Quantum Magneto-Straintronics Transport in Graphene: A Realistic Model

Lorena Reis de Lima

A Thesis

in

The Department

of

Physics

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Science (Physics) at

Concordia University

Montréal, Québec, Canada

April 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By:	**Lorena Reis de Lima**
Entitled:	**Quantum Magneto-Straintronics Transport in Graphene: A Realistic Model**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Physics)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

________________________________ Chair
Dr.

________________________________Examiner 1
Dr. Pablo Bianucci

________________________________ Examiner 2
Dr. Saurabh Maiti

________________________________ Supervisor
Dr. Alexandre Champagne

Approved by ________________________________
Dr. Valter Zazubovits, Chair
Department of Physics

Approved by ________________________________
Dr. Pascale Sicotte, Dean
Faculty of Engineering and Computer Science

Abstract

Quantum Magneto-Straintronics Transport in Graphene: A Realistic Model

Lorena Reis de Lima

The main objective of this research project is to develop an applied theoretical model to describe the quantum transport of a suspended monolayer graphene transistor under the presence of a magnetic field and unaxial strain. In the literature, we find several theoretical models for studying monolayer and multilayer graphene under several conditions and considering different physical properties such as spin. Although those proposed models predict interesting physical phenomena, such as magnetic confinement of particles, they are unrealistic and often incomplete in terms of experimental design. Here, we present a sophisticated model using a more applied approach, enabling future realistic experiments and the extraction of important experimental data regarding physical phenomena such as quantum magneto transport in strained graphene junctions, bridging theory and experiment. In order to do so, we develop a set of mathematica codes to calculate conductivity for monolayer graphene, considering both magnetic field – which contributes to the Hamiltonian as a vector potential- and the uniaxial mechanical strain (x direction) – which contributes with strain induced potentials. Two main new device geometries are introduced in this thesis: a transistor with a magnetic field applied to its channel and another with both magnetic field and unaxial mechanical strain. We report that those devices are candidates for quantum electronic components, which should be of much interest for applications. They have a high on-off ratio and their conductivity is easily suppressed by the presence of a magnetic field and mechanical strain.

Acknowledgments

Being a scientist is my childhood dream. My first real life contact with a scientist was when I was 3 years old, in the Heart Hospital's Children Center, in Sao Paulo, Brazil. I was born with a congenital heart disease named scimitar syndrome, discovered and first treated by a female scientist from the Johns Hopkins Children Center, Catherine Neil. I of course didn't know about her at the time, but I got to know a cardiologist researcher that had the same name as me, Lorena. She was one of the many responsible for saving my life. She was also one of the first inspirations I had to become a scientist - or as my high school philosophy teacher would say, a ”philosopher of nature”. Thus, to her I give the first acknowledgment of this thesis.

I also couldn't make it until here without the strong support of my family: My aunts Delcia and Maria do Socorro, who raised me and are the mothers life brought to me; my brother Artur, who colored my difficult undergraduate years with his happiness, love and kindness; my grandmother Maria, who taught me strength and persistence - although I haven't been very good learner of those lessons in the last few months.

My sincere thanks go also to the friends I made in the past years, that gave me support, love, advices and held my hand during the ups and downs of my mental health. Thank you to my friend Margaret, my favorite astrophysicist and *gringa*. Thank you to my friends from my undergraduate years: Ludmila, Felipe, Pedro, Joao Augusto, Joao Valeriano (Vava), Deborah, Vitor, Mariana, Amanda, Ranier and Tabata. Thank you to my friends that I met in Montreal: Leticia, Murilo, Rodrigo, Renata and Felipe Alves. Last but not least, thank you to my friends from UFPa: Amanda Cristina, Gustavo, Clara and Joaquim.

This thesis would never be finished without the constant support, patience, guidance and science and life advices from Alex Champagne. Alex is a extremely kind supervisor and an amazing scientist. Thank you for doing your best to keep me inspired and to go with me through all my dark and unproductive times. Thank you for giving me the opportunity of coming to Montreal and to do my master. I also take the opportunity to give a warmthul and sincere thank you to my colleagues from Alex's research group: Linxiang Huang, Israel, Amin and Wyatt (that graduated some years ago). I also would like to thank the Department of Physics from Concordia University, specially ms. Marie-Anne, prof. Pablo Bianucci, prof. Saurabh Maiti, prof. Re Mansbach, prof. Patrick Doane, mr. Wentworth Brookes and ms. Nata Zazubovits.

I also would like to thank some professors from University of Brasilia that are very important to me. Much-missed prof. Cinthia Schwantes, prof. Ivan Ferreira and prof. Qu Fanyao. I wish your time in earth had been longer and breaks my heart all of you are gone and I couldn't say goodbye. Thank you for the unconditional support and inspiration. My sincere thanks to prof. Arsen Melikyan and prof. Luis de Miranda, who both inspired my love for mathematics.

Last but not less important, I would like to thank my family from Montreal. First, my cat Lorenzo. He was literally the first friend I had in this city. Although he is extremely clueless, he is the most lovely and adorable creature that always welcomes me home and gives me reason to wake up everyday (literally, since he screams for food at 6 am). Thank you to my boyfriend Nathan, for his love, kindness, for being understanding and encouraging. Thank you to his cat Moo, who I adopted with much love. Not so much of a thank you to his spider, who tried to kill my cat (actually, Lorenzo tried to eat her).

The last but not least thank goes to myself. I used to be a curious, rebel and optmistic child/teenager. I feel like I let this fearless and dreamer version of me go and I need to get her back. I am only here because of that and I feel like self-doubt and impostor syndrome have spoiled my ability to recognize that. Thank you, Lorena, and wherever you are, I hope you come back soon, because we have a PhD to finish.

Contents

List of Figures

Chapter 1

Magneto-Transport In Strained Graphene: Why An Applied Model Is Needed?

1.1 Motivation to Bridge Theoretical and Experimental Quantum Straintronics

Straintronics in 2D materials - especially in graphene - is an exciting field to explore for many reasons, such as: tailoring energy and quantum transmission of electrons [4], induction of strong gauge fields generating pseudomagnetic quantum Hall effects and Landau Levels [5, 6, 7]; valley-polarized quantum transport [8]. In the literature, one may find a rich variety of experiments and theoretical models regarding strained graphene [9, 10]. However, when it comes to applied models that add magnetic fields [11] in strained graphene devices, we still need theories that take into account realistic physical aspects in order to bridge theory and experiment. In this project, we aim to develop a realistic applied model for an experimental device that includes both strain [1] and magnetic field [2, 12, 13]. We focus on developing an applied model that enables us to start exploring quantum-magneto-strained monolayer graphene transistors. We aim to introduce a theory that is able to show that is possible to build those type of devices and also that this technology will enable us to manufacture, in the future, better graphene transistors [14, 15, 16, 17]. For example, we want to show that this new class of transistors has a better on-off ratio, that the magnetic field and the strain combined makes it easier to control its conductivity, that the presence of the magnetic field will make it possible to build high quality graphene transistors with less mechanical strain applied, when compared with other strained transistors in the literature [1].

In one of our group's recent work [1] we explored Graphene Quantum Strain Transistors (GQST) by proposing a realistic experimental platform for uniaxial strain in ballistic graphene. The device consisted in a suspended GQST, where unaxial mechanical strain could be applied in the graphene crystal acting as the channel of the transistor. This work compiles some of the most recent efforts to control quantum transport in graphene by means of a mechanical strain. It also gives us space to think about what possible effects could be observed if we took into account not only the presence of a strain induced gauge potentials [18], but also the contribuition of a real external magnetic field applied to the sample. How easy would it be to control both quantum and magnetic transport properties? Could we observe exciting effects such as Landal Levels (LL) and the Quantum Hall Effect (QHE)?Which additions to the theory would be necessary? Those are the questions that guided and inspired the development of the present thesis.

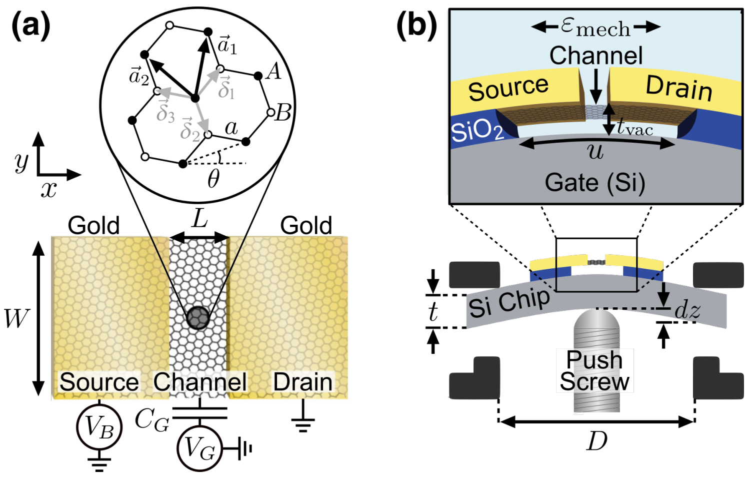


Figure 1.1: Previous work by a former member of our group, on the quantum-straintronics transport in graphene [1]. Figure 1.1(a) shows a top-view perspective of the graphene device design used to study the effects of strain on the quantum transport in graphene. Figure 1.1(b) highlights the geometrical aspects of the experiment, showing how mechanical strain is added to the suspended channel.

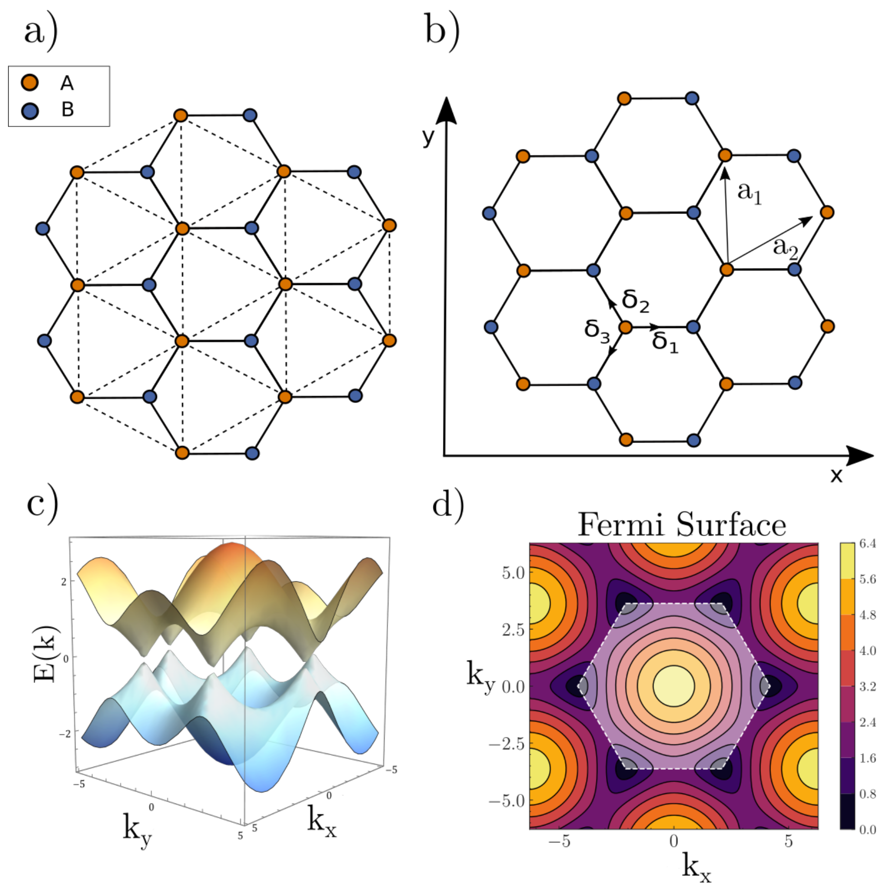


Figure 1.2: Some of the important theoretical aspects of graphene: (a) Shows its crystal structure, (b) from left to right: honeycomb lattice with emphasis on the Bravais lattice, shown in dashed (the lattice made of solid lines is the actual crystal structure). We can also visualize the cells with real space lattice vectors $\vec{a}_1, \vec{a}_2$ and nearest neighbors distances $\vec{\delta_1}$, $\vec{\delta_2}$ and $\vec{\delta_3}$.(c) Shows its energy bandgap obtained via tightbinding calculations. (d) Shows part of its lattice in the reciprocal space, focusing on its Brillouin zone.

1.2 Physics Background To Calculate Ballistic Conductivity in Graphene

1.2.1 A Brief Introduction to Graphene

Graphene is a bidimensional material made of carbon atoms. Its crystal structure has the form of a honeycomb. Its lattice has a spacing of $\sqrt{3}a$, were a is the spacing between neighboring atoms. Its basis contains two atoms per unit cell, labelled as A and B, generating two sub-lattices, as shown in figure 1.2.(a).

Graphene electronic transport can be modelled by a Dirac Equation, where the Fermi velocity plays the role of the limit velocity of charge carriers. Using a tight-binding approach, one may find its energy dispersion is:

$$E(k) = \pm|\gamma|\sqrt{3 + 2\left[\cos\left(\frac{3ak_x}{2\sqrt{3}} - \frac{ak_y}{2}\right) + \cos\left(\frac{3ak_x}{2\sqrt{3}} + \frac{ak_y}{2}\right) + \cos\left(ak_y\right)\right]} \ . \qquad (1.2.1)$$

A graphic visualization of 1.2.1 is given in Figure 1.2.(a).

In terms of the Dirac equation, the graphene Hamiltonian can be written as:

$$\hat{H} = -iv_F\hbar\vec{\sigma}\cdot\vec{\nabla} \ . \qquad (1.2.2)$$

V_F corresponds to the Fermi velocity of the charge carriers, $\hbar$ is the Planck constant and $\vec{\sigma}$ is the Dirac matrix operator.

When developing an applied theory to describe graphene transistors, Equation 1.2.2 will be the key equation for our calculations. We will modify this Hamiltonian for the problems we want to solve (for example, strained graphene), by adding the apropriate potentials which contain the information of strain, magnetic field, etc [15].

In the following Chapters, we will discuss in details how these modifications take place.

1.2.2 From Pure Theory to Applied Theory: How to Simulate Graphene Transistors?

In order to go from pure theory to a device in the real world, we have to be able to modify the graphene Dirac Equation following the correct steps and adding the correct experimental parameters to the Equation. The steps done in the present work can be divided as: writing down the correct Hamiltonian with the correct added potentials; identifying the correct boundary conditions for the case we are working on; solving the eigenvalue problem to get the eigenfunctions for each region of the device (source, channel and drain); with the eigenfunctions, solve the boundary condition problem to get an Equation for the transmission amplitude; getting the transmission probability by taking the modulus square of the transmission amplitude; summing the transmission probability of all the transmission modes to get the total conductance or conductivity.

In order to clarify our methods, below we give a detailed example on how to follow these steps.

We give the example of an unstrained graphene device, a known problem solved by Tworzydlo et al. [3].

The Hamiltonian

The Hamiltonian describing an unstrained graphene device at low energies is given by the following Equation:

$$\frac{\hbar v_F}{i}\begin{pmatrix}\sigma_x\partial_x+\sigma_y\partial_y & 0\\ 0 & \sigma_x\partial_x-\sigma_y\partial_y\end{pmatrix}\Psi = E_n\Psi \ . \tag{1.2.3}$$

This is the full Hamiltonian, meaning it takes in consideration the contribution of the wavefunctions from the *K* and *K'* Dirac points in graphene. However, the valley degeneracy allows us to rewrite the Hamiltonian in terms of only one of the symmetry points - e.g., the *K* point [3]. Thus, Equation 1.2.3 is rewritten as:

$$\hbar v_F\begin{pmatrix}0 & \hat{k}-i\hat{q}_n\\ \hat{k}+i\hat{q}_n & 0\end{pmatrix}\Psi = \hat{E}_n\Psi \ . \tag{1.2.4}$$

Here, the index n stands for the quantization of the problem, i.e., it corresponds to the $n-th$ modes, as detailed by [3]. $-i\partial_x$ was replaced by $\hat{k}$ and $-i\partial_y$ was replaced by $\hat{q}_n$.

Solving for Eigenfunctions

With Equation 1.2.4 in hands, we can now diagonalize the Hamiltonian to obtain the eigenfunctions. First, we need obtain the eigenenergies, by using the known linear algebra diagonalization method:

$$Det|(H-E_nI)| = 0 \ , \tag{1.2.5}$$

where H is given by 1.2.4, E_n is the set of eigenenergies and I is the identity matrix with the same dimensions as our Hamiltonian. Performing the calculation, we obtain:

$$\hat{E}_n = \pm\hbar\sqrt{\hat{k}^2+\hat{q}_n^2} \ . \tag{1.2.6}$$

Using Equations 1.2.4 and 1.2.6, we solve in Mathematica for the plane wave solutions of the Hamiltonian:

$$\Psi_{q_n,y,k,x} = a_n \begin{pmatrix} 1 \\ \pm\frac{k+iq_n}{\sqrt{k^2+q_n^2}} \end{pmatrix} \exp\left(iq_n y\right)\exp\left(ikx\right) \ , \tag{1.2.7}$$

Where $\hat{q}_n$ and $\hat{k}$ are, respectively, the y and x component of the linear momentum operator.

Solving the Boundary Problem: Transmission Probability and Conductance

Using the eigenfunctions obtained for the unstrained graphene Hamiltonian, we can now define the boundary condition Equations to obtain the transmission amplitude. Basically, we are solving a quantum barrier problem. We imagine the incident wavefunction coming from the left (in experimental terms, the charge carriers coming from the device's source), at some energy E_n, in the $n-th$ mode. This incident wave is attempting to go through the channel and the drain. The regions (source, channel, drain) are defined, respectively, as region I ($x < 0$), region II ($0 < x < L$) and region III ($x > L$):

$$\begin{cases} \Psi_I, & \text{for } x < L \\ \Psi_{II}, & \text{for } 0 < x < L \\ \Psi_{III}, & \text{for } x > L \end{cases} \ , \tag{1.2.8}$$

where:

$$\Psi_I = \Psi_{q_n,y,k,x} + r_n \Psi_{q_n,y,-k,x} \ , \tag{1.2.9}$$

$$\Psi_{II} = \alpha_n \Psi_{q_n,y,\tilde{k},x} + \beta_n \Psi_{q_n,y,-\tilde{k},x} \ , \tag{1.2.10}$$

$$\Psi_{III} = t_n \Psi_{q_n,y,k,x-L} \ , \tag{1.2.11}$$

The k and $\tilde{k}$ correspond, respectively, to the value of $\hat{k}$ in the scattering state incident from the left at energy E and the value of $\hat{k}$ when it reaches the $n-th$ mode. If this were the strained case, they would represent $\hat{k}$ for the unstrained and unstrained case, and $\tilde{q}n = q_n - A_y$, were A_y is the strain induced potential.

The parameters r_n, t_n, α_n and β_n correspond, respectively, to the reflection and transmission coefficients and constants associated with the normalization of the wavefunction. L is the length of the device, usually of the order of 10^2 nm.

Using the continuity of the wavefunction,

$$\Psi_I\Big|_{x=0} = \Psi_{II}\Big|_{x=o} \ , \tag{1.2.12}$$

$$\Psi_{II}\Big|_{x=L} = \Psi_{III}\Big|_{x=L} \ , \tag{1.2.13}$$

and solving the two Equations above for t_n, the transmission amplitude is found as:

$$\frac{k\tilde{k}}{k\tilde{k}\cos{(\tilde{k}L)} - i(-q_n^2 + \sqrt{(k^2+q_n^2)(\tilde{k}^2+q_n^2)})\sin{(\tilde{k}L)}} \ . \tag{1.2.14}$$

Using the fact that $T_n = |t_n|^2$, the transmission probability is given by:

$$\frac{k^2\tilde{k}^2}{k^2\tilde{k}^2\cos{(\tilde{k}L)}^2 + (-q_n^2 + \sqrt{(k^2+q_n^2)(\tilde{k}^2+q_n^2)})^2\sin{(\tilde{k}L)}^2} \ . \tag{1.2.15}$$

After the transmission probability is found, one can get the conductance G, by summing over all of the possible transmission modes:

$$G = g_0 \sum_{n=0}^{N_{max}-1} T_n \ , \tag{1.2.16}$$

where $g_0 = \dfrac{4e^2}{h}$. The conductivity is given by:

$$\sigma = G \times \frac{L}{W} \ . \tag{1.2.17}$$

In Equation 1.2.16, N_{max} corresponds to the maximum number of transmission modes, set by contact doping. This quantity is given by:

$$N_{max} = \frac{|E_F|}{\hbar v_F}\frac{W}{\pi} + \frac{1}{2} \ . \tag{1.2.18}$$

The parameters E_F and W correspond, respectively, to the Fermi energy and the width of the device.

1.3 Experimental Parameters for Modeling Quantum Magneto-Straintronics Transport in Graphene

Our main goal is to be able to calculate the G (or also σ) in our simulated devices, given a set of tunable parameters. For the case of a monolayer graphene transistor that is suspended and strained, the tunnable parameters are gate voltage (V_G) and mechanical strain (ϵ_{mech}). When the magnetic field is added, the field is also one of the tunnable parameters. In this section, we describe the parameters needed to build our theory.

1.3.1 Device Parameters

Length (L) and Width (W)

The dimensions of the graphene chip are very important parameters and sometimes can be limited by the theory. Here, we refer to the length L as the x-dimension of the graphene channel in our transistor. In real devices produced by our group, L is usually of the order of 90 nm. For computational simulation, this real value can be used, but also replaced by some value of the order of 100 nm, in order to reduce the running time of the code.

The width W refers to the width of the channel, in experiments it is usually around 640 nm, but in simulations, it can be fixed at 1000 nm, in order to facilitate calculations.

Fermi level (μ_{CH}), Strains (ϵ, ϵ_0) and Gate Capacitance (c_G)

The Fermi level in the channel (μ_{CH}) is one of the most important parameters that enables us to study the charge conductance of the device. This potential is usually adjusted by the value of gate voltage. It also depends on the charge doping from impurities present in the device channel (n_{imp}). How much we can tune μ_{CH} will be dictated by the gate capacitance (c_G), which affects the density of charge carriers due to electrostatic gating (n_{tot}). If there is no strain and $n_{imp} = 0$, μ_{CH}, n_{tot} and c_G are given, respectively, by:

$$\mu_{CH} = \hbar v_F \sqrt{\pi Abs[n_{tot}]} \ , \tag{1.3.1}$$

$$n_{tot} = \frac{c_G}{e}(V_G - V_D) \ , \tag{1.3.2}$$

$$c_G = \frac{\epsilon_{vac}\epsilon_{SiO_2}}{d_{SiO_2}\epsilon_{vac} + d_{vac}\epsilon_{SiO_2}} \ . \tag{1.3.3}$$

$\epsilon_{vac} = 8.85 \times 10^{-12}$ F/m and $\epsilon_{SiO_2} = 3.9 \times 8.85 \times 10^{-12}$ F/m are respectively the permittivity of vacuum and the permittivity of SiO_2, while $d_{vac} = 200$ nm and $d_{SiO2} = 100$ nm in experiments. For simulation purposes, we usually assume the silicon oxide thickness is 0 (to maximize the energy range) and that the thickness of the vacuum is around 50 nm. With those values, we get $c_G = 1.7 \times 10^{-4}$ Farad/m^2.

The gate voltage range is usually between -15 V and 15 V. Later in this thesis you will see that this range can slightly change due to some restrictions in the model, or because we needed to reduce the code running time, and other details which will be later introduced.

In the Mathematica code, we use the contact doping to be of the order of 0.1 eV (or 0.087 eV). However this value can vary. Our previous experiments show that $\mu_{CH} \approx 55$ meV [19].

When strain is applied to the device, the Equation for n_{tot} will change. The strain will enter the Hamiltonian as an scalar potential, originated both from the ϵ_{mech} applied to the sample and $\epsilon_{thermal}$, that is a built in strain due to thermal contraction. This latter one remains constant during the experiments.

Usually, the total strain ϵ_{tot} ranges from 0% to 3%. As we will see later in this thesis, because of the presence of the magnetic field, only a small amount of strain will be needed to control conductivity. In experiments, $\epsilon_{thermal} \approx 1.3\%$, however in our codes we usually consider it to be zero since, at the moment, it is not within the scope of this work to deal with the thermal contributions.

Given those details, when considering strain, n_{tot} becomes:

$$n_{tot} = \frac{c_G}{e}(V_G - V_D) + sign[g_\epsilon]\frac{g\epsilon(1-\nu)(\epsilon+\epsilon_0)^2}{\pi(\hbar v_F)^2} \ , \tag{1.3.4}$$

where ν is the Poisson ratio [20] and g_ϵ is the change in the work function per percent of strain [21].

1.3.2 Instrumentation Parameters

The Mechanical Strain (ϵ_{mech})

The mechanical strain will depend on several experimental parameters, such as the dimensions of the sample, the clamps put on the device, how much the sample is bent by the means of the push screw. When the screw is pushed (z-direction), the sample will be stretched in the x-direction, i.e., the gold clamps will move apart by a certain dx (see Figure 1.1). This dx is given by:

$$\Delta x = \frac{3ut}{D^2}\Delta z \ , \tag{1.3.5}$$

where $D = 8.18nm$, u is the suspension length and t is the thickness of the SiO_2 wafer. Also,

$$\epsilon_{mech} = \frac{\Delta x}{L} \ . \tag{1.3.6}$$

The Magnetic Field

The magnet used in Champagne's laboratory are made of Helmoltz superconducting coils, enabling us to produce large fields. The sample is placed inside the cilyndrical bore (interior of the magnet), and is subjected to a uniform magnetic field perpendicular to the sample (z-direction). The magnet can produce a field up to 9 T.

In terms of calculations, the magnetic field enters the Hamiltonian as a vector potential. For our simulations, due to theory limitations that will be explained later, we were limited to fields up to 2.03 T.

What Does the Code to Get the Conductivity Looks Like?

With all of these important paramaters and theoretical background, a code can finally be written. The code language is chosen to be Mathematica. The code is actually divided in three Mathematica notebooks: "Parameters", "Transmission Equation" and "Conductivity". The *Parameters* notebook is filled with all the relevant experimental parameters and physical constants. It also defines arrays to store data for conductivity, Fermi energy, gate voltage and other parameters related to the potentials added to the hamiltonian (for example, mechanical strain and magnetic field). The *Transmission Equation* notebook solves the boundary condition problem to get the transmission amplitude, taking its modulus square to get the mode transmission, and then exports the simplified version of the equation to a mathematica file that is going to be read in the conductivity code. The *Conductivity* code uses the parameters and the mode transmission to calculate the transmission over all modes (conductance), the conductivity (Equation 1.2.17), Fermi energy, the strain, the magnetic field and in some cases the Fano factor. A flowchart of the code is shown in Figure 1.3. Inside the *for* loops,

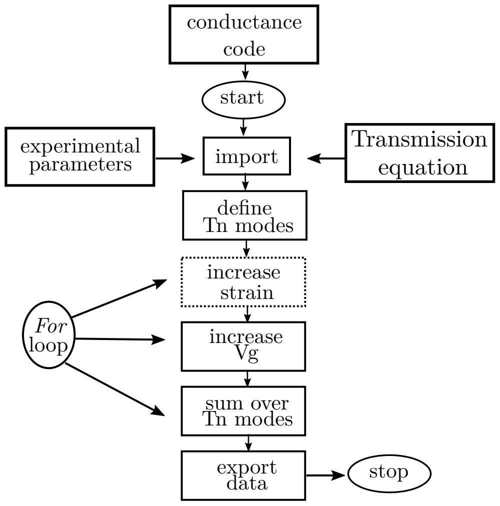


Figure 1.3: Flowchart to depict the structure of the code developed to calculate transport properties for monolayer graphene transistors. The dashed box indicates where to add loops for quantities such as strain and magnetic field.

quantities such as the density of states, Fermi energy in the contacts, charge carriers density, vector potentials along y-direction in each Dirac cone, amongst other quantities, are calculated, in order to get the transmission in terms of these experimental parameters. This is essentially what makes the simulation closer to real world experiments.

All the codes are available in this thesis, in the Appendix A. Mathematica files of the codes can be provided by the authors upon reasonable request.

1.3.3 Thesis Structure

This thesis is organized in 4 Chapters. The first Chapter introduces graphene, the motivations of the work, gives an outlook of the state of the art in terms of an applied theory to model ballistic transport in graphene, discuss the main parameters (device and instrumentation) necessary to build this theory and the step-by-step general guide to calculate conductivity.

In Chapter 2, we start from a theory to describe the magneto-transport of Dirac fermions in graphene, and we expand that theory by adding experimental parameters to it, in order to calculate conductance.

In Chapter 3, we combine the magneto-transport theory developed in Chapter 2 with the strain-tunable transport theory developed previously by [1]: we add strain to the magneto-transport. This chapter is dedicated to introduce an applied theory to describe the quantum-magneto-strain transport in a graphene transistor.

In Chapter 4, we give an outlook of the impact of this thesis and give some ideas on how to continue this project. We also highlight the main results obtained in this research.

In the Appendix A, we provide the Mathematica notebooks in PDF format. We added them in order of utilization and we will refer to them in the apropriate chapters and sections.

Chapter 2

Modelling Ballistic Magneto-Transport Experiments

In this Chapter, we add magnetic field to the graphene Hamiltonian, for a graphene transistor with magnetic field in the channel of the device. We solve for the wavefunctions in each region (source, channel, drain), solve the quantum barrier problem with the apropriate boundary conditions, get the transmission expression and use it to calculate the conductance of the transistor. This is a new contribution, since there was no previous work, to the best of our knowledge, which did this kind of calculation for an applied device including the parameters introduced in Chapter 1.

We also verify that this magneto-transport calculation agrees with the previous transport results when there is no magnetic field [3, 1, 22]. Finally, we show some simulation results for conductivity and we discuss the data obtained.

2.1 Applied Theory for Quantum-Magneto Graphene Transistors

2.1.1 Deriving the Hamiltonian for Graphene in a Magnetic Field

In this problem, we have a monolayer graphene device, where the channel is submited to a static $\hat{z}$ magnetic field (see Figure 2.1), which is oriented perpendicular to the graphene plane [2]. However, $B = 0$ in the source and in the drain of the simulated device. We then have a quantum magnetic barrier to solve, similar to the standard problems of quantum barriers we find in quantum

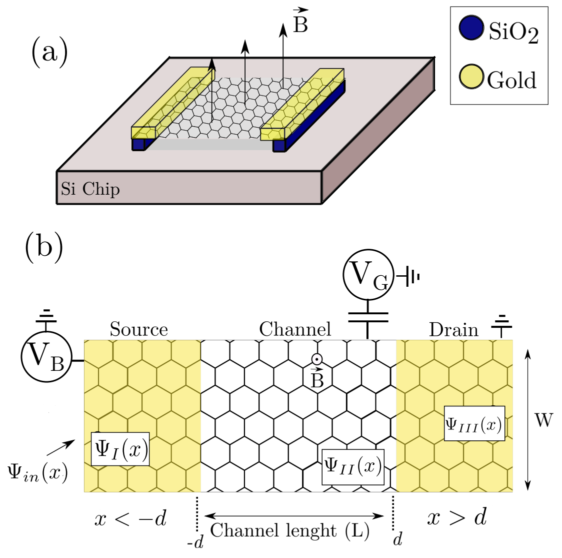


Figure 2.1: Cartoon depicting the device developed by merging [2] theory and our applied model. (a) Shows a representation of how the graphene transistor would look like. (b) Highlights the physical parameters used to describe the model. $\Psi_{in}(x)$ is the incident electronic wavefunction that comes from the source. The other wavefunctions are the eigenfunctions of the graphene Hamiltonian in each region of the device. The gold leads were drawn transparent to emphasize that the electrons are injected into the channel from graphene contacts from the gold directly.

mechanics textbooks. A sketch of the experimental device is shown in Figure 2.1. For simplicity, we do not take into account the electronic spin degree of freedom. We model the problem using the Dirac equation formalism, as shown in the Equation 2.1.1:

$$\vec{\sigma} \cdot [\vec{p} + \frac{e}{c}\vec{A}(x,y)]\Psi(x,y) = E\Psi(x,y) \ . \tag{2.1.1}$$

The $\vec{\sigma}$ represents the Pauli Matrices – in this problem, we take into consideration only $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$. The eigenvectors $\Psi(x,y)$ are spinors, given by:

$$\Psi(x,y) = \begin{pmatrix} \Psi_+ \\ \Psi_- \end{pmatrix} \ . \tag{2.1.2}$$

Using the second Landau gauge, we write the vector potential as

$$\vec{A}(x,y) = A(x)\hat{y} \ , \tag{2.1.3}$$

with the following relation between the magnetic field and the vector potential:

$$A(x) = B_x\hat{y} \ . \tag{2.1.4}$$

Given that the transverse momentum $\vec{p}_y$ is conserved and using the Pauli Matrices, we can rewrite Equation 2.1.1 as the following set of coupled Equations:

$$[\partial_x \pm p_y \pm \frac{e}{c}A(x)]\Psi_\pm(x) = iE\Psi_\mp(x) \ . \tag{2.1.5}$$

We can write this system of coupled Equations as a 1 dimensional Schrödinger decoupled Equation:

$$\left[\partial_x^2 - V_\pm(x) + \epsilon^2\right]\Psi_\pm(x) = 0 \ , \tag{2.1.6}$$

where

$$V_\pm(x) = \pm\frac{e}{c}\partial_x A(x) + \left[p_y + \frac{e}{c}A(x)\right]^2 \ . \tag{2.1.7}$$

2.1.2 Deriving the Transmission Equation

To solve the previous equation, we consider that there is only magnetic field within the chanel of the device, i.e.,

$$\vec{B} = \begin{cases} B_0\hat{z}, & \text{for} - d \leq x \leq d \\ 0, & \text{otherwise.} \end{cases} \tag{2.1.8}$$

It is convenient then to write the magnetic field as a Heaviside step function:

$$B(x,y) = B_0\theta(d^2 - x^2) \ . \tag{2.1.9}$$

We can also write an expression for the vector potential for each region of the device: the source ($x < -d$), the drain ($x > d$) and the channel ($|x| \leq d$), as the following:

$$A(x) = \frac{c}{\hbar l_B^2} \begin{cases} -d, & \text{for } x < -d \text{ (Region I)} \\ x, & \text{for } |x| \leq d \text{ (Region II)} \\ d, & \text{for } x > d \text{ (Region III)} \end{cases} , \tag{2.1.10}$$

where $l_B \equiv \sqrt{\dfrac{\hbar}{eB_0}}$ is the magnetic length.

The electronlike incident wavefunction that enters the device via source, up to the normalization constant, has the following format:

$$\Psi_{in}(x) = \begin{pmatrix} 1 \\ \dfrac{p_x + i(p_y - d/l_B^2)}{|\mathbf{p}|} \end{pmatrix} \exp ip_x x \ , \tag{2.1.11}$$

where $|\mathbf{p}|$ is a normalization that includes the normalization of the wavefunction and the normalization of the **p** momentum vector. Notice that there is a shift in the momentum in the y direction. This happens because of the Landau gauge we used.

Let ϕ and ϕ' be, respectively, the incident angle and the scattering angle for the wavefunction. Thus, it is natural to chose to rewrite our momenta in this new parametrization. Thus, we get:

$$p_x = \epsilon \cos\phi, \quad p_y = \epsilon \sin\phi + \frac{d}{l_B^2} \ , \tag{2.1.12}$$

$$p_x^{'} = \epsilon \cos\phi', \quad p_y^{'} = \epsilon \sin\phi' - \frac{d}{l_B^2} \ , \tag{2.1.13}$$

where we found $\sin\phi'$ using the fact that $p_y = p_y'$, thus

$$\sin\phi' = \frac{2d}{\epsilon l_B^2} + \sin\phi \ . \tag{2.1.14}$$

In Figure 2.1.b, we can visualize better the physical problem we are trying to solve.

Plugging condition for region I from 2.1.10 into 2.1.7 we obtain the following differential Equation:

$$\left[\partial_x^2 - p_y^2 + 2p_y\frac{d}{l_B^2} - \frac{d}{l_B^4} + \epsilon^2\right]\Psi_\pm(x) = 0 \ . \tag{2.1.15}$$

Proceeding in the same fashion for regions II and III, we obtain, respectively:

$$\left[\partial_x^2 \mp \frac{1}{l_B^2} - p_y^2 - 2p_y\frac{x}{l_B^2} - \frac{x^2}{l_B^4} + \epsilon^2\right]\Psi_\pm(x) = 0 \ , \tag{2.1.16}$$

$$\left[\partial_x^2 - p_y^2 - 2p_y\frac{d}{l_B^2} - \frac{d^2}{l_B^4} + \epsilon^2\right]\Psi_\pm(x) = 0 \ . \tag{2.1.17}$$

The scattering state $\Psi_{in}(x)$ can be written for the source, channel and drain as the following:

$$\begin{cases} \Psi_I, & \text{for } x < -d \\ \Psi_{II}, & \text{for } |x| \leq d \\ \Psi_{III}, & \text{for } x > d \end{cases} \ , \tag{2.1.18}$$

where:

$$\Psi_I = \Psi_{q_n,y,k,x} + r_n\Psi_{q_n,y,-k,x} \ , \tag{2.1.19}$$

$$\Psi_{II} = \alpha_n\Psi_{q_n,y,k,x} + \beta_n\Psi_{q_n,y,-k,x} \ , \tag{2.1.20}$$

$$\Psi_{III} = t_n\Psi_{q_n,y,k,x_1-x_2} \tag{2.1.21}$$

where:

$$\Psi_I(x) = \begin{pmatrix} 1 \\ \exp(i\phi) \end{pmatrix} \exp(ip_x x) + r \begin{pmatrix} 1 \\ -\exp(-i\phi) \end{pmatrix} \exp(-ip_x x) \ , \tag{2.1.22}$$

$$\Psi_{II}(x) = \Sigma_\pm c_\pm \begin{pmatrix} D_{(\epsilon l_B)^2/2-1}[\pm\sqrt{2}(x/l_B) + p_y l_B] \\ \pm i\frac{\sqrt{2}}{\epsilon l_B} D_{(\epsilon l_B)^2/2}[\pm\sqrt{2}(x/l_B) + p_y l_B] \end{pmatrix} \tag{2.1.23}$$

and

$$\Psi_{III}(x) = t\sqrt{p_x/p'_x}\begin{pmatrix} 1 \\ \exp(i\phi') \end{pmatrix}\exp(ip'_x x) \ , \tag{2.1.24}$$

are the eigenfunctions of the Schroedinger Equations 2.1.15-2.1.17.

In Equation 2.1.23, D_ν are parabolic cylinder functions [23], a type of special functions that arise in solutions of some type of differential Equations.

By solving the Boundary Condition problem, i.e., the system of Equations

$$\Psi_I(q_n, y, k, -d) = \Psi_{II}(q_n, y, k, -d) \tag{2.1.25}$$

and

$$\Psi_{II}(q_n, y, k, d) = \Psi_{III}(q_n, y, k, d) \ , \tag{2.1.26}$$

we get the transmission amplitude:

$$t = \frac{2i\epsilon l_B\sqrt{2p'_x/p_x}\cos(\phi)}{\exp[I(p_x + p'_x)d]\mathcal{D}}(u_2^+ v_2^- + v_2^+ u_2^-) \ . \tag{2.1.27}$$

The Operator $\mathcal{D}$ is given by

$$\begin{aligned}\mathcal{D} = (\epsilon l_B)^2 \exp[I(\phi' - \phi)](u_2^+ v_2^- - v_2^+ u_2^-) - 2(v_1^+ v_2^- - v_2^+ v_1^-) \\ + I\sqrt{2}\epsilon l_B[\exp(I\phi')(v_1^+ u_2^- + u_2^+ v_1^-) + \exp(-I\phi)(u_1^+ v_2^- - v_2^+ u_1^-)] \ ,\end{aligned} \tag{2.1.28}$$

where

$$u_1^\pm = D_{(\epsilon l_B)^2/2-1}[\pm\sqrt{2}(-d/l_B) + p_y l_B] \ , \tag{2.1.29}$$

$$v_1^\pm = D_{(\epsilon l_B)^2/2}[\pm\sqrt{2}(-d/l_B) + p_y l_B] \ , \tag{2.1.30}$$

$$u_2^\pm = D_{(\epsilon l_B)^2/2-1}[\pm\sqrt{2}(d/l_B) + p_y l_B] \ , \tag{2.1.31}$$

$$v_2^\pm = D_{(\epsilon l_B)^2/2}[\pm\sqrt{2}(d/l_B) + p_y l_B] \ . \tag{2.1.32}$$

We obtain the transmission probability by taking the modulus squared of the transmission amplitude.

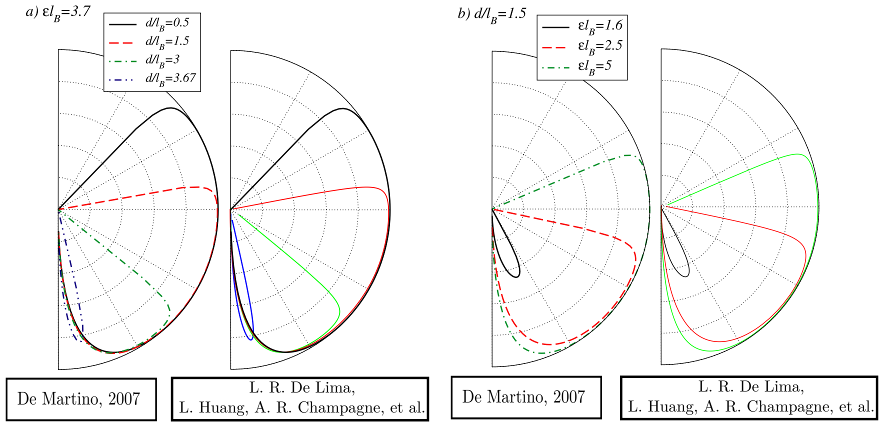


Figure 2.2: Comparison between transmission probability graphs obtained by [2] (graphs in the left) and by our code (graphs in the right). We use the same colors as the original author for better visualization. The transmission probabilities are calculated in angles inside the interval $\left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$.

Because the scattering angle ϕ' can only be calculated with Equation 2.1.14, this implies that for certain incident angles, there will be no transmission. All incident angles will lead to reflected states if

$$k_F l_B \leq d/l_B \quad . \tag{2.1.33}$$

In Figure 2.2, we reproduce some of the possible transmission probabilites given this condition, from reference [2].

2.1.3 The Code

The code for calculating the magneto-transport properties of the simulated device introduced here is very similar to the code introduced in Chapter 1. However, instead of doing a *for* loop over the strain, here the loop is done for the magnetic field. For now, we focus on simulations for constant values of the magnetic field; this will be generalized later.

After importing the parameters and the transmission expression, the code starts by defining the number of transmission modes, set by the contact doping. We also define l_B, the ratio d/l_B and the vector potential along the y-direction, as defined in Equation 2.1.10.

The next step is to increase gate voltage. Inside this loop, we calculate the carrier density from gate and scalar potential, the total carrier density, the Fermi level in the channel and $e/l_B =$

$\sqrt{\pi Abs[n_{tot}]}l_B$. Those are all quantities that depend on the gate voltage and, as explained in Chapter 1, are co-dependant with each other.

Inside the same loop, we create an array to save the values of gate voltage, channel and Fermi level potentials. With the values inside this array, we can then create another for loop to sum over all transmission modes, with the quantization for y-component of momentum as:

$$q_n = \frac{\pi}{W}\left(N_q - \frac{1}{2}\right) + A_y \ , \tag{2.1.34}$$

where A_y is the vector potential along the y-direction, given by $A_y = -\frac{e}{\hbar}\frac{L}{2}B$ and N_q is the number of allowed transmission modes, calculated in the code. The scattering angle ϕ is now calculated as:

$$\phi = ArcSin\left(\frac{q_n}{k_{Fcontact}}\right) \ , \tag{2.1.35}$$

where $k_{F\ \ contact} = \mu_{CH}/\hbar v_F$.

After summing over the transmission modes, we get the conductance, then we calculate the conductivity and create a data file with the values of G and E_F for each value of V_G for a fixed value of B. A schematics of the code is shown in Figure 2.3.

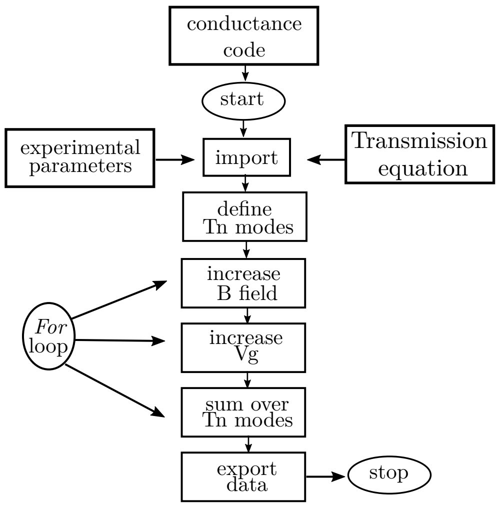


Figure 2.3: Flowchart of the code used to calculate the transport properties of a graphene transistor under a constant magnetic field.

2.1.4 Quick Guide on How to Use the Code

The Mathematica notebooks can be found at the Appendix A. To run the code, please first run "*DeMartino_export_Tn_expression.nb*". This will generate a .mx file with the analytic expression for the transmission equation (way too big and complicated to be written in the thesis). After that, run "*Gr_Vars_Params_21_09_2023.nb*". This will load all the physical constants and experimental parameters necessary to run the transport code.

Finally, run "*Conductivity_DeMartino_B_VG_ac.nb*". This will generate a data file with the data mentioned in the above paragraph. You can plot the data or analyze it using the software of your preference; we used Igor Pro 7.

2.2 Simulation Results and Predictions

First we needed to show that our code converges to reproduce the results present in the literature [3] in the limit when B approaches to zero. This result is shown in the graph bellow.

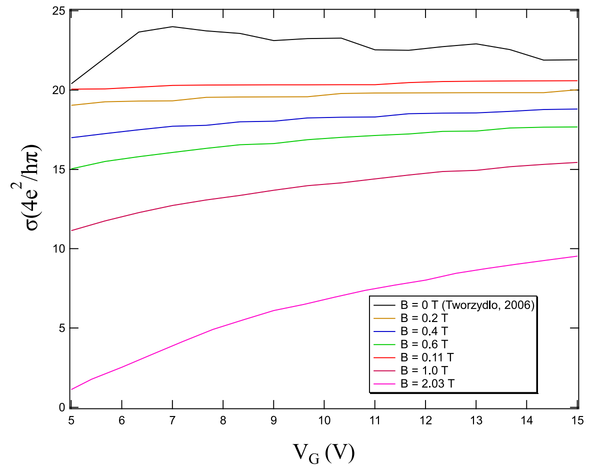


Figure 2.4: Graph showing conductance for different values of decreasing magnetic field in order to show that at low values of B, the model converges to a model with no strain and no magnetic field, predicted in [3]. The simulation results for the case with no B field is possible thanks to the work of Linxiang Huang and Andrew McRae.

As shown in Figure 2.4, when the value of the magnetic field is very close to zero ($B = 0.11$ T), the conductivity (red curve) converges to the case shown in the literature (black curve). For computational reasons, we cannot set the value of the magnetic field exactly to zero, but it is enough to see that there is a convergence pattern.

The reader may notice that we only plot values up to $B = 2.03$ T. This because the condition $k_F l_B \leq d/l_B$ imposes severe restrictions on the parameters we can use for our devices. For L = 100 nm, $l_B = 32$ nm, $k_F = 10^8 m^{-1}$, we can only have a maximum value of $B = 0.6$ T. This is a low value of field if one is interested in effects like the quantum Hall or Landau levels. Thus, this is out of the scope of this thesis, i.e., we did not expect to see the quantum Hall effects after analysing the consequences that this condition imposes to our theory.

To overcome this problem and at least try to have a higher value for the magnetic field, we changed some parameters. We set $L = 80$ nm, $W = 1000$ nm, $u = 2000$ nm (suspension length). We chosen a value of landau Level filling fraction $\nu = 18$, so that we could hope that with those dimensions for the device and that filling fraction would help us to find a higher value of magnetic field allowed by our theory. Since

$$\nu = \frac{n_c}{n_f} = \frac{density\ of\ charge\ carriers\ per\ unit\ area}{density\ of\ quanta\ of\ magnetic\ flux\ per\ unit\ area}\ , \tag{2.2.1}$$

where, for zero strain

$$n_c = \frac{c_{BG}}{e}(V_G - V_D)\ , \tag{2.2.2}$$

and

$$n_f = \frac{BS}{\frac{h}{e}S}\ , \tag{2.2.3}$$

where S is the area of the device. Using those Equations, $V_G - V_D = 5$V, and the known parameters introduced in Chapter 1, we get $B = 2.03$ T.

2.2.1 Suppression of Conductance due to a Magnetic Field

Many properties can dictate how effective a transistor is, but one of the most important aspects is related to the function of the component itself. Transistors's main function is the amplification of the voltage signal. This also enables the device to work as a switch. In order to be a high quality transistor, one must manufacture devices that good on/off ratios [24]. It is shown in literature [25, 26, 8] that many carbon based devices, such as nanoribbons, that can be used as efficient conductivity switches. Adding temperature to the picture also affects suppression; however, exploring the thermal conductivity suppression is out of the scope of this work.

Some signature of suppression in the conductivity of our magnetic graphene transistor is already visible in Figure 2.4. The conductivity already drops by one order of magnitude, when the magnetic field is applied.

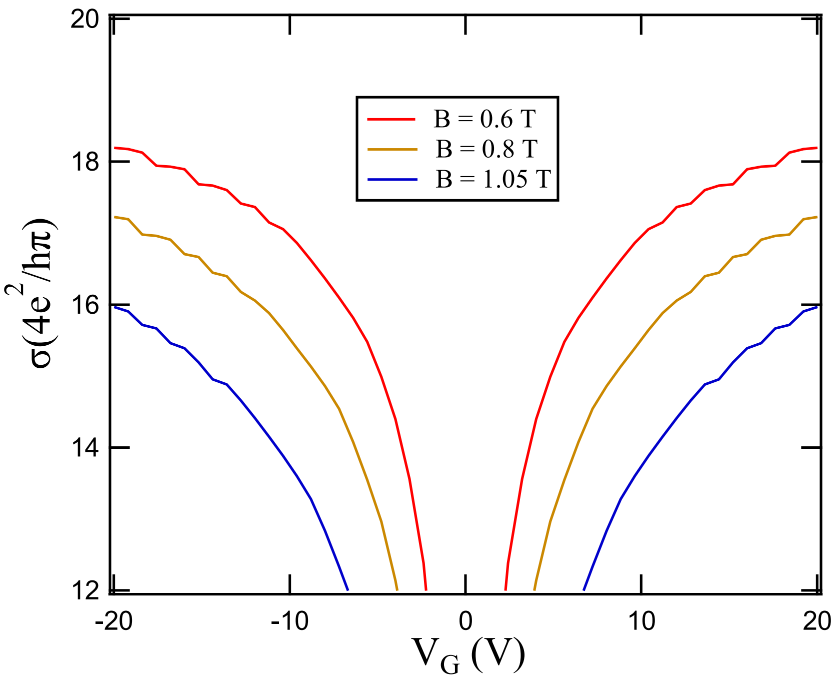


Figure 2.5: Conductivity signatures for a transistor whose channel is subjected to a uniform perpendicular magnetic field. Each curve corresponds to different fixed values for the magnetic field. The gate voltage ranges from -20 V to 20 V.

Another example of decrease in the conductivity is shown in Figure 2.5. There is also a suppression of the opening of the transmission modes in the extremity of each curve. This opening of modes occur because the quantum levels allowed are rapidly occupied by the electrons, causing the remaining electrons to abruptly jump to another level, thus showing this ladder-like pattern in the conductivity curves.

Chapter 3

Modelling Strained Magneto-Transport in Ballistic Graphene Experiments

In this Chapter, we begin by introducing the model developed in [1], for the case where the transistor is submitted to a unaxial mechanical strain and no magnetic field.

In the sequence, we introduce a new model, where we modify the Hamiltonian for monolayer graphene in the presence of a magnetic field (Equation 2.1.6) by adding a term in the potential, that takes into account the contribution of the mechanical strain. We show a step-by-step derivation of the transmission equation, which is later used to calculate the conductance in the device. We study the effects of strain and magnetic field variations on the conductance. We show that the transistors simulated with this applied theory reach an on-off ratio of the order of 10^4 at a constant voltage of 10 V, even for moderate ϵ and B.

3.1 Strained Monolayer Graphene Transistor

In Chapter 2, we introduced the model to study transport in a graphene transistor under a magnetic field. In this section, we will first introduce a model for graphene with no magnetic field, but subjected to an unaxial mechanical strain, developed in [1]. Figure 1.1 shows a schematics of the device. The Hamiltonian that describes this device is given by:

$$H = \hbar v_F[I + (1-\beta)\epsilon] \cdot \sigma \cdot (\tilde{k} - A_i) + \Delta\mu_G + \Delta\mu_\epsilon \ . \tag{3.1.1}$$

$\tilde{k}$ is the wave vector in the strained channel, σ is the set of Pauli matrices in x and y, A_i is the strain vector potentials along the y-direction in the dirac cones (where i = 1,2,3). $\Delta\mu_G$ and $\Delta\mu_\epsilon$ are respectively the gate potential and the strain scalar potential. A full detailed derivation of the transmission Equation is available in [1]. The main objective of revisiting this model is show what the Hamiltonian with strain looks like and what are some of the effects of adding the strain [21]. The first interesting result is how the Dirac cones are affected by the scalar potential $\Delta\mu_\epsilon$ in the Hamiltonian.

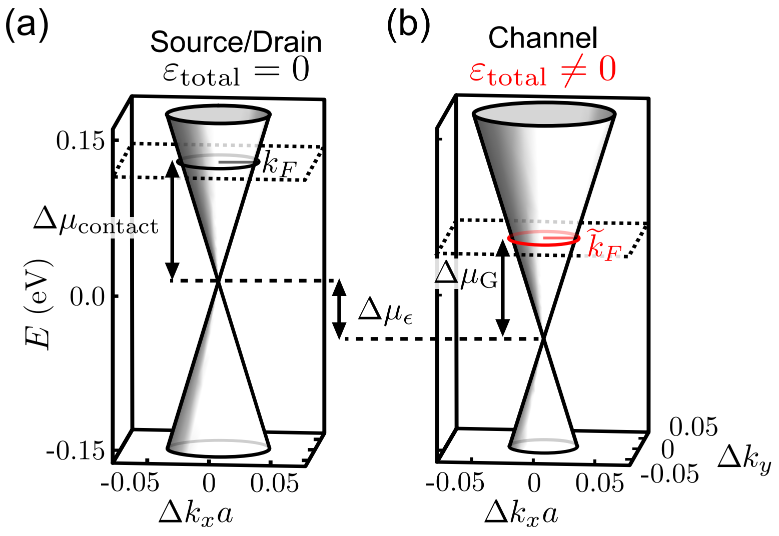


Figure 3.1: Cartoon deptcing how strain affects the Dirac cones [1].

In Figure 3.1.(a), we show the Dirac cone and Fermi circle in the unstrained case; 3.1.(b) shows the strained case. Note that the presence of the potential due to strain shifts the Fermi circle from the Dirac cones. The Fermi wave vector in the source/drain contacts now also depends on both contact doping and on the strain-induced potential.

The strain-induced vector potentials (A_i) also play a very important role. Due to the applied mechanical strain, there is a shift in the conductivity curves. The bigger the strain, the more suppressed the conductivity will be. This opens a discussion as to whether or not we will be able to enhance this suppression when adding a magnetic field, thus creating even better transistors.

3.2 Hamiltonian for Strained Monolayer Graphene in the Presence of a Magnetic Field and Strain

3.2.1 Adding the Contribution of Strain Potentials to the Magnetic Hamiltonian

We modify the Hamiltonian in Equation 2.1.6 to add mechanical strain. A cartoon of the idealized device is shown in Figure 3.2.

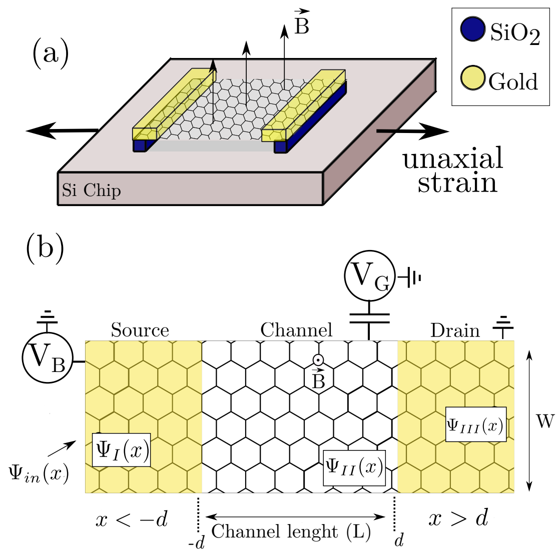


Figure 3.2: Cartoon depicting the device developed with our applied model, combining unaxial mechanical strain and magnetic field in the channel. (a) Shows a representation of how the graphene transistor would look like. (b) Highlights the physical parameters used to describe the model. $\Psi_{in}(x)$ is the incident electronic wavefunction that comes from the source. The other wavefunctions are the eigenfunctions of the graphene Hamiltonian in each region of the device. The gold leads were drawn transparent to emphasize that the electrons are injected from the graphene contacts, not from the gold.

In Equation 2.1.6, we will add the potential in Equation 2.1.7, where $A(x)$ will now also include the mechanical strain vector in the y direction A_i, thus our new differential Equation will look like

$$\left[\partial_x^2 \mp \left(\frac{1}{l_B^2}\right) - p_y^2 - 2p_y\left(\frac{x}{l_B^2} + \frac{e}{c}A_0\right) - \left(\frac{x^2}{l_B^4} - \frac{2xA_0}{l_B^2} + A_0^2\right) + \epsilon^2\right]\Psi_\pm = 0 \ , \tag{3.2.1}$$

where

$$A(x) = \frac{xc}{el_B^2} - A_0 \ , \tag{3.2.2}$$

and

$$A_0 = \frac{\hbar}{e} A_i \ . \tag{3.2.3}$$

Equation 3.2.1 may be very complicated to solve, so we have to figure out a way to make this Equation more elegant in terms of solution. We can do that by introducing a new variable – x_0 – in Equation 3.2.2,

$$x_0 = A_0 \frac{el_B^2}{c} \ . \tag{3.2.4}$$

Thus, we can rewrite Equation 3.2.2 as:

$$A(x) = \frac{(x - x_0)c}{el_B^2} \ . \tag{3.2.5}$$

Since $A_{i,y}$ is different in each valley, so x_0 will be. Because of the valley symmetry, we only need to define three values for $A_{i,y}$, those being defined in the following Equation:

$$A_{1,y}(\theta) = \frac{4\pi(\epsilon + \epsilon_0)}{3\sqrt{3}a} \nu \sin(\theta) + \frac{\beta(\epsilon + \epsilon_0)(1 + \nu)}{2a} \sin(3\theta) \ , \tag{3.2.6}$$

$$A_{2,y}(\theta) = \frac{2\pi(\epsilon + \epsilon_0)}{3a} \bigg(-\frac{1}{\sqrt{3}} \nu \sin(\theta) + \nu \cos(\theta) \bigg) + \frac{\beta(\epsilon + \epsilon_0)(1 + \nu)}{2a} \sin(3\theta) \ , \tag{3.2.7}$$

$$A_{3,y}(\theta) = \frac{2\pi(\epsilon + \epsilon_0)}{3a} \bigg(-\frac{1}{\sqrt{3}} \nu \sin(\theta) - \nu \cos(\theta) \bigg) + \frac{\beta(\epsilon + \epsilon_0)(1 + \nu)}{2a} \sin(3\theta) \ . \tag{3.2.8}$$

In Equations 3.2.6, 3.2.7 and 3.2.8, $\nu = 0.165$, $\beta = 2.5$, $a = 1.42 \times 10^{-10}$ m and $\theta = 15 \times \frac{\pi}{6}$ are, respectively, the Poisson ratio in graphene, the electron-phonon coupling, the lattice constant and the crystal angle with respect to the x-axis.

Finally, setting $x_2 = x - x_0$ we rewrite our decoupled set of differential Equations as

$$\bigg[\partial_x^2 \mp \frac{1}{l_B^2} - p_y^2 - 2p_y \frac{x_2}{l_B^2} - \frac{x_2^2}{l_B^4} + \epsilon^2 \bigg] \Psi_\pm = 0 \ . \tag{3.2.9}$$

In the operators defined by the parabolic cylinder functions, the only change will be a shift in the

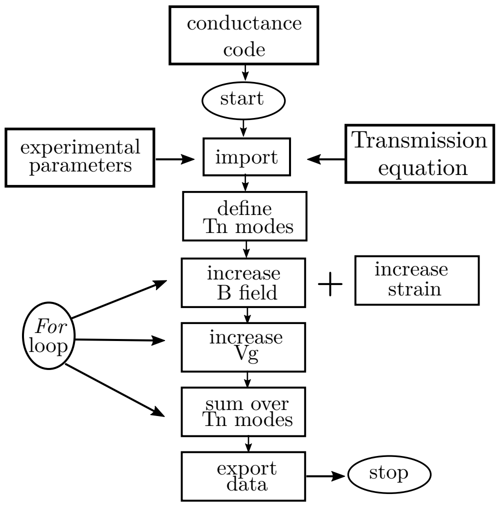


Figure 3.3: Flowchart of the code used to calculate G in a graphene transistor subjected to a perpendicular magnetic field on its channel and to an unaxial strain.

d of each Equation. For u_1 and v_1 (Equations 2.1.29 and 2.1.30), $\frac{-d}{l_B} \rightarrow \frac{(-d - x_0)}{l_B}$. For u_2 and v_2 (Equations 2.1.31 and 2.1.32), $\frac{d}{l_B} \rightarrow \frac{(d - x_0)}{l_B}$. Also, since we are using SI units, $c \rightarrow \hbar$ in Equation 3.2.9. To simplify those shifts in the calculations, we call the shift on u_1 and v_1 dl_1 and the ones in u_2 and v_2 as dl_2 in our code. Thus, in our calculations, the only change will be rewritting Equations 2.1.28 and 2.1.27 taking these changes into account.

3.2.2 The Code

Adding strain to the picture not only increases the level of complexity of the mathematical equations, but also increase the computational time to run the code. In order to keep the running times compatible with the time available, we had to make some adaptations. First, we broke the problem in two codes: one code to simulate the transport properties of the transistor when the value of the strain is fixed, but the value of the magnetic field changes and another code to simulate the case where the magnetic field is fixed, and the strain is varying. We also kept the number of steps for magnetic field and strain to 50 points. The magnetic field maximum value is set up to 2.03 T, the limit value for our simulated device's parameters.

Figure 3.3 shows a schematics of the code. The codes initially do the same as the previous ones: start by importing both the parameters notebook and the transmission expression file. The differences in the other steps of the code are explained bellow.

When varying the strain at a fixed B, the number of transmission modes set by contact doping is defined. Then, we write a *for* loop to increase the magnetic field. Inside this loop, we first define the B increments, l_B, the minimum value of strain (imported from the parameters notebook) and the magnetic vector potentials along y direction. Different from the case where we only had magnetic field, now we must add the change in each $A_{i,y}$ (only three of them are considered, because of the symmetry of the valleys), because of the mechanical induced strain. Because the potential will depend in the value of the vector potential induced by the strain, as established in Equation 3.2.5, x will need to be calculated in each valley with the correspondent changes in $A_{i,y}$.

After defining those parameters in the code, we write – still inside the strain loop – another *for* loop, but this time to increase the gate voltage. Inside of it, the quantities that directly depend on V_G (carrier density from gate and scalar potential, total carrier density, Fermi level in the channel, $e/l_B = \sqrt{\pi Abs[n_{tot}]l_B}$ are calculated. The arrays defined in the paramaters notebook for current strain, gate and channel Fermi level are then used to store the values calculated in the loop.

Following that, we define an array to store T_n. Inside it, we write another *for* loop, this time to sum over all the transmission modes. In each valley, one mode is calculated. The incident angle ϕ is also calculated for each valley, since they depend on the q_n associated to the vector strain potential for each case. Finally, the average transmission for all six Dirac cones is calculated. However, we only keep the positive T_n values – those are the values that will matter in experiments. With that, the conductance is calculated.

After that, the procedure is similar to the previous codes. The code generate a data file with the values for gate voltage, conductivity, magnetic field, strain and Fermi Energy. The code can also be found the Appendix A, in the end of this thesis.

For the case where we vary the magnetic field at a fixed value of strain, the only difference will be that instead of a *for* loop for strain, we will write a loop for the magnetic field. The other programming steps remain virtually the same.

3.3 Results

3.3.1 Conductivity Suppression Under Fixed Magnetic Field and Varying Strain

We already know that strain can suppress the conductivity [1]. Is the behavior maintained when we add magnetic field to the picture? We found out that this is the case. We also observe that due to the combination of mechanical strain and magnetic field, this suppression occurs at a faster pace. This shows us that transport properties can be fine tuned and controlled in graphene transistors under the strained-magneto-transport regime we present. This result is shown in Figure 3.4. In this simulation, a constant magnetic field of 1 T is applied to the channel of a graphene transistor. A mechanical strain is then applied, and it increases from 0% to 1.25 %, in a gate voltage range between 5 V and 15 V. The device dimensions are $L = 80$ nm and $W = 100$ nm.

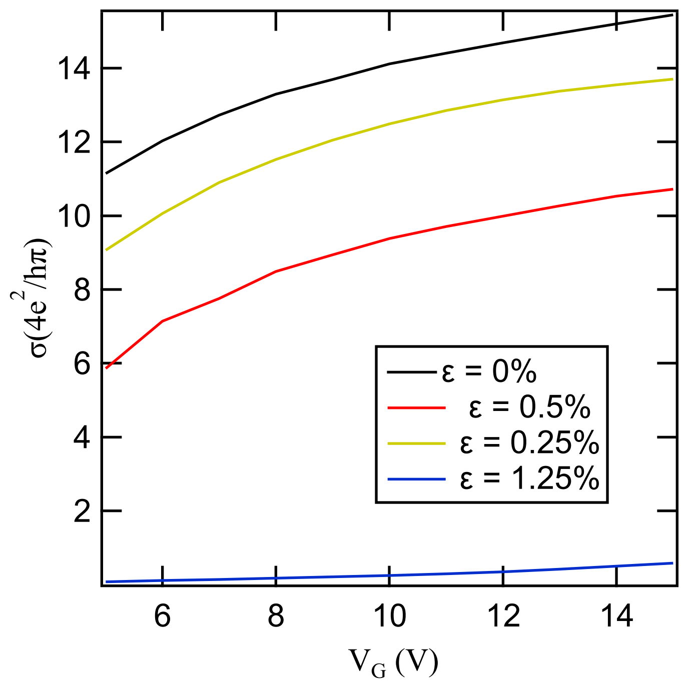


Figure 3.4: Suppression of conductivity due to strain at a constant magnetic field $B = 1$ T. A small quantity of strain (from 0% to 1.25%) was enough to reduce the conductivity to zero.

Note that, when $\epsilon = 1.25$ %, the conductivity in Figure 3.4 already drops to almost zero. This shows that this transistor, designed to work under a magnetic field and unaxial mechanical strain, is a candidate for a high on/off ratio transistor.

3.3.2 Linear Change in Conductivity Under Constant Strain and Varying Magnetic Field

Keeping the same dimensions for the device, the gate voltage range and the contact doping the same as mentioned in the previous section, we now fix the strain to 1% and vary the values of the magnetic field up to $B = 1.20$ T. The result of the simulation is shown in Figure 3.5.

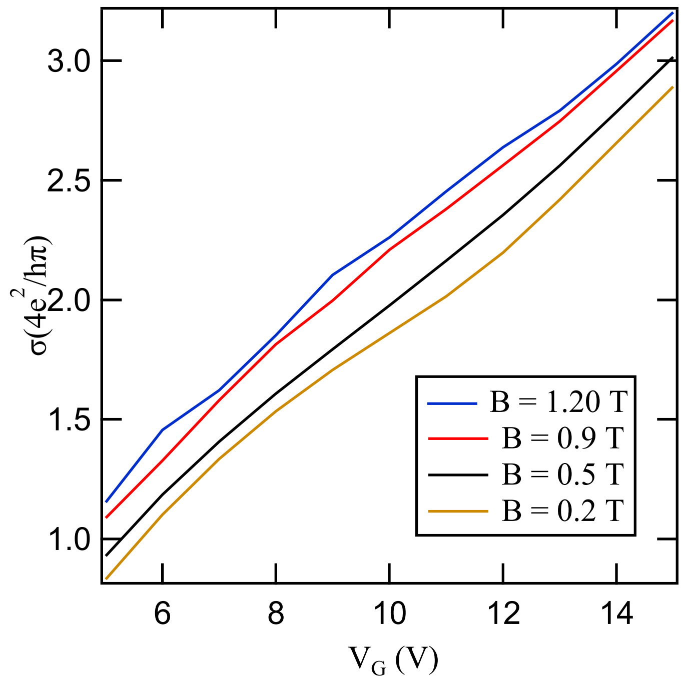


Figure 3.5: G - V_G at various values of B, with strain $\epsilon = 1\%$.

Note that the conductivity signature is mostly linear, with no signs of opening of modes. It seems that, under the presence of a magnetic field, the conductivity becomes directly dependent of the mechanical strain applied.

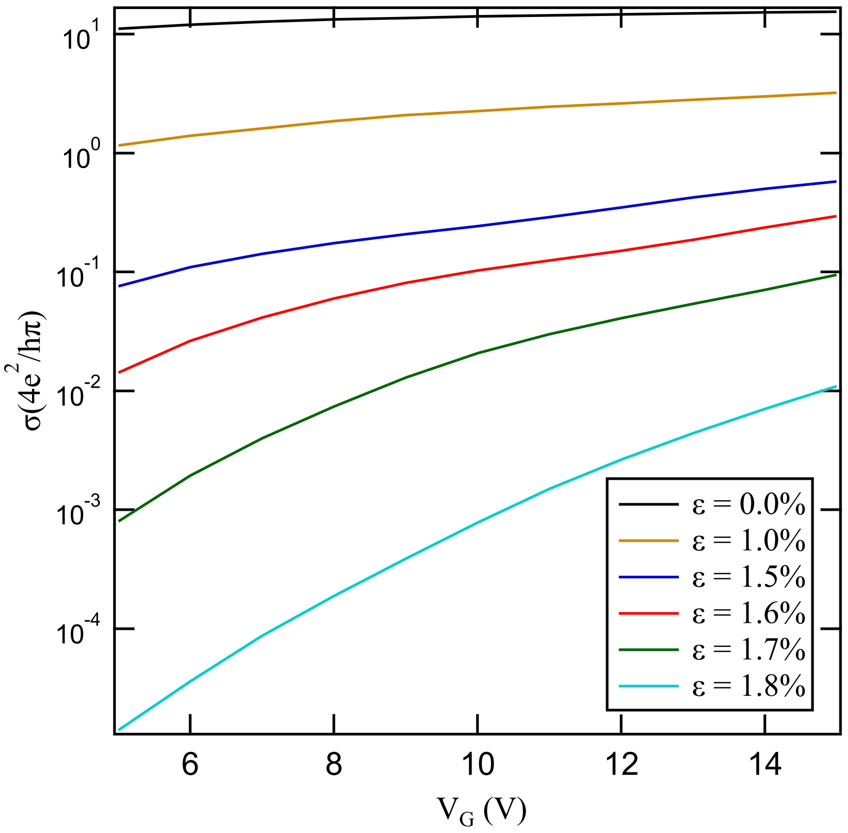


Figure 3.6: *Log(G)* curves for different values of strain at $B = 1$T.

3.3.3 On-off Ratio

The on-off ratio of a transistor is a very important indicator of the qualitity of a transistor. This value represents the ratio between the current in the *on* state and the current in the *off* state. The higher the on-off ratio, the better is the device performance, meaning that there is a low current leakage [27] [28] [29]. To achieve a $R/R_0 \geq 10^4$, the transistor with only strain described in [1] needed $\epsilon = 2.6\%$, both at $\theta = 15 \times \dfrac{\pi}{6}$ and $\mu_{contact} = 0.087$ eV. The device we propose in this Chapter (i.e., combining both mechanical strain and magnetic field in the channel) reaches the same value of on-off ratio around $\epsilon = 1.8\%$, roughly half of the strain necessary for the strained device.

Figures 3.6, 3.7 and 3.8, show, respectively, the different orders of magnitude of the conductivity values for different values of strain, the log scale version and the on-off ratio of $R/R_0 = 10^4$ at $V_G = \pm 15$ V. This was obtained by defining $V_{on} = 15$ V and $V_{off} = 5$ V. Then, we pick the

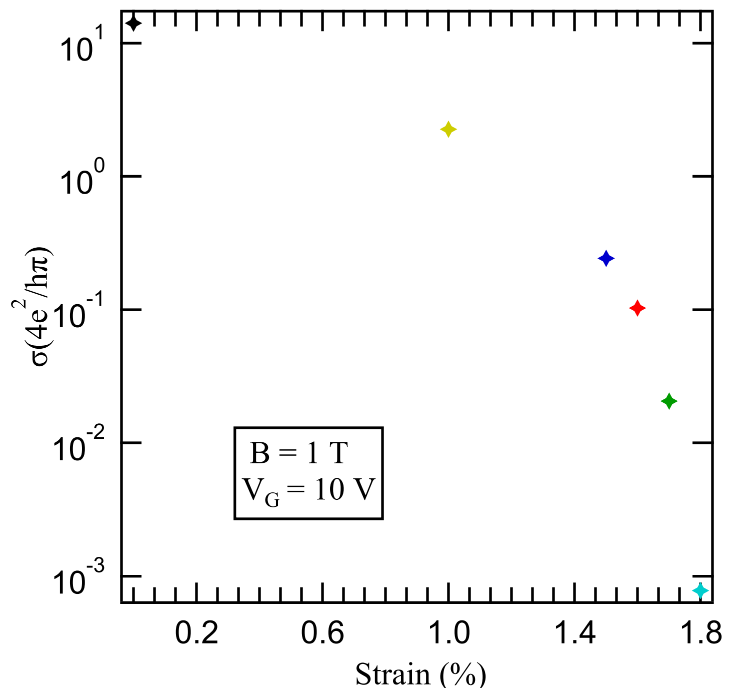


Figure 3.7: $Log(\sigma) - \epsilon$ for a quantum-magneto-strained graphene transistor under fixed values of magnetic field and vate voltage.

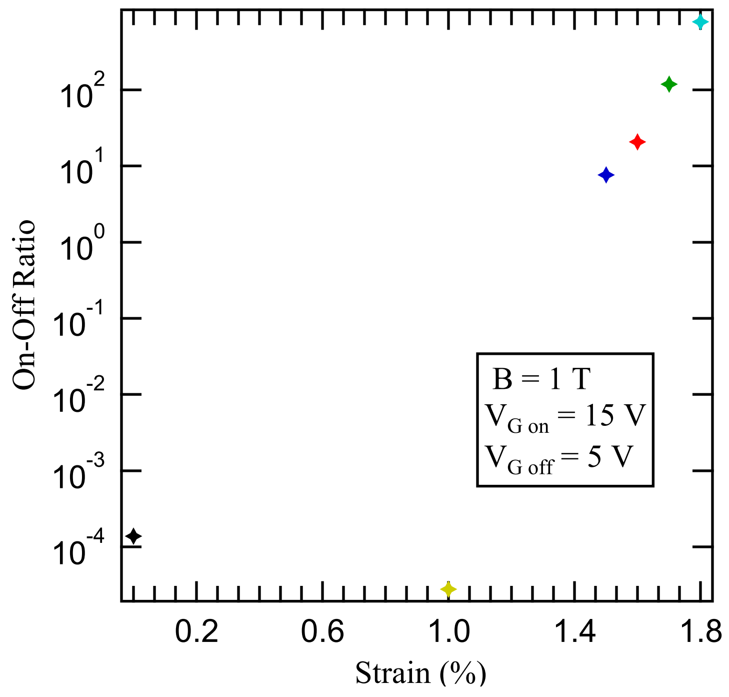


Figure 3.8: On-off ratio for a quantum-magneto-strained graphene transistor under fixed values of magnetic field and vate voltage.

values of conductivity for V_{on} and V_{off} for each curve and calculate $\frac{G_{V_{on}}}{G_{V_{off}}}$. With those values, we produced Figure 3.8.

Chapter 4

Outlook and Conclusions

4.1 Main Results, Next Steps and Expected Impact

In this thesis, we introduced an applied theory to study and model the magneto-quantum-strain transport in monolayer graphene transistors. We expanded and enhanced the theory developed in [2], by adding a magnetic field in the channel of the graphene device. By adding the experimental parameters to the picture , considering a suspended sample and by calculating experimental quantities such as the conductance and the conductivity, we enabled this work to be used to model future experiments. We also showed that in the limit where B approaches zero, we return to the previous theory developed by our group, where the Hamiltonian contains only strain.

In addition, we showed that the conductivity in these devices can be fine tuned when magnetic field and strain are combined. Our theory only contains magnetic field and mechanical strain in the chanel of the graphene device, however, it should be straight forward to add magnetic field in the contacts, as shown in [12]. In order to do so, it is easier to consider the problem as a multi-barrier quantum well. Thus, we would find new solutions for the Hamiltonian shown in Equation 2.1.6 in the contacts (source and drain). Solving this problem will require a method called the transfer matrix method, which makes it easier to find the transmission equation. After finding the transmission equation, the steps to calculate conductivity remains roughly the same (i.e., summing over all the possible modes).

Other interesting scenarios can also be predicted by using our method, given the proper adaptations. For example, one may want to reproduce our work, but this time using a bilayer graphene suspended sample, or even a twisted bilayer graphene. The graphene itself could be replaced by another two-dimensional material (thus replacing the Hamiltonian). Although the Hamiltonian will of course be changed, the transfer method matrix works for either cases, and the structure of our code, given the apropriate adaptations (for example, changing the physical parameters to the other material parameters), will still work. By using the WKB method, one could also explore other regimes of magnetic field in the transistor.

The possibilities are numerous, and there is always room for improvement, but still we expect this work to help both experimentalists and theoreticians to have a more realistic applied theory to make predictions and understand the results collected in the laboratory.

Appendix A

Mathematica Codes

In this short Appendix, we introduce the code files designed to run the simulations described in this thesis. We divided the codes between appendix sections, so the reader can follow the codes in order of appearence in the text.

In the first Section, we introduced the codes used in Chapter 2 to test if Equations 2.1.22, 2.1.23 and 2.1.24 were solutions to Equation 2.1.6, given the potential in Equation 2.1.10. The three first Mathematica codes perform this task. The fourth code in this first Section is used to derive the transmission amplitude expression given in Equation 2.1.27, by solving the boundary condition problem established in Equations 2.1.25 and 2.1.26. The last code in this Section is used to plot the transmission polar graphs, given in Figure 2.2.

In the second Section of this Appendix, we introduce the codes used to calculate the conductivity in a graphene transistor under a magnetic field, as presented in Chapter 2. The first code of the second section is the Parameters Notebook, where we define all the constants necessary to run our simulation. The second code is the Transmission Notebook and it is responsible for generating the transmission expression and exporting to an external file. After running those two codes, we then run the third code, which contains all the equations necessary to calculate the conductivity in graphene. This third code generates a data file that can be read with Igor Pro, and thus be used to plot graphs or to do data analysis.

In the third Section of this Appendix, we introduce the codes used to run the simulations in Chapter 3. The first two codes of this Section contain the Parameters Notebook used to define all the physical constants and parameters used to simulate a graphene transistor under magnetic field and strain. The third code exports the transmission expression. The fourth and the fifth codes are used to simulate a graphene transistor for two different cases: when the magnetic field is at a fixed value and when the strain is at a fixed value.

To run the codes, please always follow this order: 1) Parameters Notebook, 2) Export Transmission Expression, 3) Conductivity Code.

A.1 Codes Used In Chapter 2 - Transmission Plots

A.1.1 Codes to Test Eigenfunctions Solutions

Testing solution in Region I:

```
(*Test: Region I*)
         região   unidade
(*Region I, A(x)=0*)
   região   unidade imaginá

Remove["Global`*"]
remove

In[ ]:= d[x_] = D[#, {x, 2}] &; (*dx^2 = dx*dx but has to be applied on the function*)
                derivada
vecA = x * (c / (e * lb^2));
ν = (((eps * lb)^2) / 2);
z = Sqrt[2] * ((x / lb) + (py * lb));
    raiz quadrada

In[ ]:= (*Define wavefunction expression for region I in matrix form
                                            unidade imaginária
 ( i tested to see if our operator works in that form and yes,
  it matches the calculations i did by hand as well)*)
(*Wavefunctions Ψ₊=Ψp, Ψ₋=Ψm*)
ΨI = {{Exp[I * px * x]}, {Exp[I * ϕ] * Exp[I * px * x]}} +
      ex… unidade imagin… ex… unidad… ex… unidade imaginária
   {{r * Exp[-I * px * x]}, {-1 * r * Exp[-I * ϕ] * Exp[-I * px * x]}};
       exp… unidade imaginária      exp… unidad… exp… unidade imaginári
```

Equation 3 expanded form : $\left[(\partial_x)^2 \mp \frac{e}{c}\,\partial_x A(x) - {p_y}^2 - \frac{2e}{c}\,p_y A(x) - \frac{e^2}{c^2}\,A(x)^2 + \epsilon^2\right]\psi_\pm = 0$

Equation 3 original form : $\left[(\partial_x)^2 \pm \frac{e}{c}\,\partial_x A(x) + \left(p_y + \frac{e}{c}\,A(x)\right)^2 + \epsilon^2\right]\psi_\pm = 0$

** ** ** ** ** ** Equation 3 with $A(x) =$

0 : $\left[(\partial_x)^2 - {p_y}^2 + \epsilon^2\right]\psi_\pm = 0$ ----> test bellow is testing this line ** ** ** ** ** ** **

Equation 3 with $A(x) = -d$ (Region I, Ψ_I) :

** Alex Notes **

(1) there is *a* sign error above in , Equation 3 with $A(x) = 0$ should actually be :

$\left[(\partial_x)^2 - {p_y}^2 + \epsilon^2\right]\psi_\pm = 0$

(2) there is an error in the wavefunction. Note ! that $\psi_\pm$ is only defined ! for Region II,

(the + − mean left and right moving, but in region one we only have *a* right moving wavefunction !)

Also from Equation (9) $\psi_I = \text{Exp}[I * \text{px} * x] * \{1, \text{Exp}[I * \phi]\} + r * \text{Exp}[-I * \text{px} * x] * \{1, -\text{Exp}[-I * \phi]\}$

(3) Please use this info and try to check : $\left[(\partial_x)^2 - {p_y}^2 + \epsilon^2\right]\psi_I = 0$

```
(*test each part of equation 3 with A=
 0 separately to see if it matches handwritten calculations*)
d[x][ΨI, x]
```

```
Out[ ]= {{-e^(i px x) px^2}, {-e^(i px x+i ϕ) px^2}}
```

$\text{Out[ ]=}\ \left\{\left\{-e^{i\,\text{px}\,x}\,\text{px}^2\right\}, \left\{-e^{i\,\text{px}\,x + i\,\phi}\,\text{px}^2\right\}\right\}$

```
py^2 * ΨI
```

Out[ ]= $\{\{e^{i\,px\,x}\,py^2\},\ \{e^{i\,px\,x+i\,\phi}\,py^2\}\}$

```
eps^2 * ΨI
```

Out[ ]= $\{\{e^{i\,px\,x}\,eps^2\},\ \{e^{i\,px\,x+i\,\phi}\,eps^2\}\}$

In[ ]:=
```
d[x][ΨI, x] - py^2 * ΨI + eps^2 * ΨI // FullSimplify // MatrixForm
                                        simplifica complet···  forma de matriz
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -e^{-i\,px\,x}\left(-eps^2+px^2+py^2\right)\left(e^{2\,i\,px\,x}+r\right) \\ -e^{-i\,(px\,x+\phi)}\left(-eps^2+px^2+py^2\right)\left(e^{2\,i\,(px\,x+\phi)}-r\right) \end{pmatrix}$$

In[ ]:=
```
(*use the parametrization for px and py that De Martino gives*)
px = eps * Cos[ϕ]
           cosseno
(*py=eps*Sin[ϕ]+d/lb^2*)
           seno
py = eps * Sin[ϕ]
           seno
```

Out[ ]= eps Cos[ϕ]

Out[ ]= eps Sin[ϕ]

In[ ]:=
```
d[x][ΨI, x] - py^2 * ΨI + eps^2 * ΨI // FullSimplify // MatrixForm
                                        simplifica complet···  forma de matriz
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
(*Region I, A(x) = - d*)
  região   unidade imaginária
```

Equation 3 with $A(x) = -d$:

$$\left[(\partial_x)^2 - p_y{}^2 + 2\,p_y\,\frac{d}{l_B{}^2} - \frac{d^2}{l_B{}^4} + \epsilon^2\right]\psi_\pm = 0$$

In[ ]:=
```
Remove["Global`*"]
remove
```

In[ ]:=
```
d[x_] = D[#, {x, 2}] &; (*dx^2 = dx*dx but has to be applied on the function*)
        derivada
```

In[ ]:=
```
(*Wavefunction in region I *)
                         unidade imaginária
ΨI = {{Exp[I * px * x]}, {Exp[I * ϕ] * Exp[I * px * x]}} +
      ex··· unidade imagin··  ex··· unidad··  ex··· unidade imaginária
    {{r * Exp[-I * px * x]}, {-1 * r * Exp[-I * ϕ] * Exp[-I * px * x]}};
          exp···  unidade imaginária          exp···  unidad··· exp···  unidade imaginári

(*calculate equation*)
```

In[ ]:= $d[x][\Psi I, x] - py^2 * \Psi I + 2 * py * \frac{d}{lb^2} * \Psi I - \frac{d^2}{lb^4} * \Psi I + eps^2 * \Psi I$ // FullSimplify // MatrixForm
simplifica complet⋯ forma de matriz

Out[ ]//MatrixForm=

$$\begin{pmatrix} -\frac{e^{-i\,px\,x}\left(lb^4\left(-eps^2+px^2\right)+\left(d-lb^2\,py\right)^2\right)\left(e^{2\,i\,px\,x}+r\right)}{lb^4} \\ -\frac{e^{-i\,(px\,x+\phi)}\left(lb^4\left(-eps^2+px^2\right)+\left(d-lb^2\,py\right)^2\right)\left(e^{2\,i\,(px\,x+\phi)}-r\right)}{lb^4} \end{pmatrix}$$

In[ ]:= (*use the parametrization for px and py that De Martino gives*)

$px = eps * Cos[\phi]$
cosseno

$py = eps * Sin[\phi] + d / lb^2$
seno

Out[ ]= $eps\,Cos[\phi]$

Out[ ]= $\frac{d}{lb^2} + eps\,Sin[\phi]$

In[ ]:= (*calculate equations again*)

$d[x][\Psi I, x] - py^2 * \Psi I + 2 * py * \frac{d}{lb^2} * \Psi I - \frac{d^2}{lb^4} * \Psi I + eps^2 * \Psi I$ // FullSimplify // MatrixForm
simplifica complet⋯ forma de matri

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Testing solution in Region II:

Equation 3 in De Martino paper, with $A(x) = \frac{xc}{e l_B{}^2}$:

$$\left(\partial_x{}^2 \mp \frac{1}{l_B{}^2} - p_y{}^2 - \frac{2 p_y x}{l_B{}^2} - \frac{x^2}{l_B{}^4} + \epsilon^2\right)\Psi_\pm = 0$$

In[ ]:=
```
Remove["Global`*"]
```
remove

••• Remove: There are no symbols matching "Global`*".

In[ ]:=
```
(*define hamiltonian in equation (3). I don't put
```
unidade imaginária
```
 the energy² here because it enters later in the DSolve*)
```
resolve equação diferencial

```
(*Define "H+"*)
```
$$\texttt{H1[f\_] = Dt[f, \{x, 2\}]} - \frac{1}{\texttt{lb}^2} * \texttt{f} - \texttt{py}^2 * \texttt{f} - \left(\frac{2 * \texttt{py} * \texttt{x}}{\texttt{lb}^2}\right) * \texttt{f} - \left(\frac{\texttt{x}^2}{\texttt{lb}^4}\right) * \texttt{f}$$
derivada total

Out[ ]=
$$-\frac{f}{lb^2} - f\, py^2 - \frac{2 f\, py\, x}{lb^2} - \frac{f\, x^2}{lb^4} + \mathrm{Dt}[f, \{x, 2\}]$$

In[ ]:=
```
(*write DSolve to solve the equation*)
```
resolve equação diferencial
```
(*get Ψ+*)
DiracEq = DSolve[H1[Ψ+[x]] == -(ϵ[μ] * ϵ[μ]) * Ψ+[x], Ψ+[x], x]
```
resolve equação diferencial

Out[ ]=
$$\left\{\left\{\Psi_+[x] \to c_2\, \mathrm{ParabolicCylinderD}\left[-\frac{1}{2}\, lb^2\, \epsilon[\mu]^2,\ i\left(\sqrt{2}\, lb\, py + \frac{\sqrt{2}\, x}{lb}\right)\right] + c_1\, \mathrm{ParabolicCylinderD}\left[\frac{1}{2}\left(-2 + lb^2\, \epsilon[\mu]^2\right),\ \sqrt{2}\, lb\, py + \frac{\sqrt{2}\, x}{lb}\right]\right\}\right\}$$

```
(*write Ψ+ in matrix form*)
```

In[ ]:=
```
(*Define "H-"*)
```
$$\texttt{H2[f\_] = Dt[f, \{x, 2\}]} + \frac{1}{\texttt{lb}^2} * \texttt{f} - \texttt{py}^2 * \texttt{f} - \left(\frac{2 * \texttt{py} * \texttt{x}}{\texttt{lb}^2}\right) * \texttt{f} - \left(\frac{\texttt{x}^2}{\texttt{lb}^4}\right) * \texttt{f}$$
derivada total

Out[ ]=
$$\frac{f}{lb^2} - f\, py^2 - \frac{2 f\, py\, x}{lb^2} - \frac{f\, x^2}{lb^4} + \mathrm{Dt}[f, \{x, 2\}]$$

In[ ]:=
```
(*get Ψ-*)
DiracEq = DSolve[H2[Ψ-[x]] == -(ϵ[ν] * ϵ[ν]) * Ψ-[x], Ψ-[x], x]
```
resolve equação diferencial

Out[ ]=
$$\left\{\left\{\Psi_-[x] \to c_1\, \mathrm{ParabolicCylinderD}\left[\frac{1}{2}\, lb^2\, \epsilon[\nu]^2,\ \sqrt{2}\, lb\, py + \frac{\sqrt{2}\, x}{lb}\right] + c_2\, \mathrm{ParabolicCylinderD}\left[\frac{1}{2}\left(-2 - lb^2\, \epsilon[\nu]^2\right),\ i\left(\sqrt{2}\, lb\, py + \frac{\sqrt{2}\, x}{lb}\right)\right]\right\}\right\}$$

Testing solution in Region III:

```
(*Region III, A(x)=0*)
  región

In[]:=
Remove["Global`*"]
remove

In[]:= d[x_] = D[#, {x, 2}] &; (*dx^2 = dx*dx but has to be applied on the function*)
          derivada
(*vecA=x*(c/(e*lb^2)); → GENERAL FORM OF POTENTIAL VECTOR*)
(*ν= (((eps*lb)^2)/2); → FOR PARABOLIC CYLINDER IN REGION II*)
(*z=Sqrt[2]*((x/lb)+(py*lb)); → FOR PARABOLIC CYLINDER IN REGION II *)
     raiz quadrada

In[]:= (*Define wavefunction expression for region III in matrix form *)
ΨIII = t * Sqrt[px/px1] {{1}, {Exp[I * ϕ1]}} * Exp[I * px1 * x];
             raiz quadrada      ex⋯ unidade im⋯ ex⋯ unidade imaginária
```

Equation 3 expanded form : $\left[(\partial_x)^2 \mp \frac{e}{c}\,\partial_x A(x) - {p_y}^2 - \frac{2e}{c}\,p_y\,A(x) - \frac{e^2}{c^2}\,A(x)^2 + \epsilon^2\right]\psi_\pm = 0$

Equation 3 original form : $\left[(\partial_x)^2 \pm \frac{e}{c}\,\partial_x A(x) + \left(p_y + \frac{e}{c}\,A(x)\right)^2 + \epsilon^2\right]\psi_\pm = 0$

** ** ** ** ** ** Equation 3 with $A(x) =$

0 : $\left[(\partial_x)^2 - {p_y}^2 + \epsilon^2\right]\psi_\pm = 0$ ----> test bellow is testing this line ** ** ** ** ** **

```
(*test each part of equation 3 with A=
 0 separately to see if it matches handwritten calculations*)

In[]:= d[x][ΨIII, x]
```

Out[]= $\left\{\left\{-e^{i\,\mathrm{px1}\,x}\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\,\mathrm{px1}^2\,t\right\},\left\{-e^{i\,\mathrm{px1}\,x+i\,\phi1}\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\,\mathrm{px1}^2\,t\right\}\right\}$

```
In[]:= py^2 * ΨIII
```

Out[]= $\left\{\left\{e^{i\,\mathrm{px1}\,x}\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\,\mathrm{py}^2\,t\right\},\left\{e^{i\,\mathrm{px1}\,x+i\,\phi1}\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\,\mathrm{py}^2\,t\right\}\right\}$

```
In[]:= eps^2 * ΨIII
```

Out[]= $\left\{\left\{e^{i\,\mathrm{px1}\,x}\,\mathrm{eps}^2\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\,t\right\},\left\{e^{i\,\mathrm{px1}\,x+i\,\phi1}\,\mathrm{eps}^2\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\,t\right\}\right\}$

```
In[]:= d[x][ΨIII, x] - py^2 * ΨIII + eps^2 * ΨIII // FullSimplify // MatrixForm
                                                   simplifica complet⋯  forma de matriz
```

Out[]//MatrixForm=

$$\begin{pmatrix} -e^{i\,\mathrm{px1}\,x}\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\left(-\mathrm{eps}^2+\mathrm{px1}^2+\mathrm{py}^2\right)t \\ -e^{i\,(\mathrm{px1}\,x+\phi1)}\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\left(-\mathrm{eps}^2+\mathrm{px1}^2+\mathrm{py}^2\right)t \end{pmatrix}$$

```
In[•]:= (*use the parametrization for px and py that De Martino gives, but when x = -d*)
        px = eps * Cos[ϕ];
                   cosseno
        (*py=eps*Sin[ϕ]+d/lb^2*)
                   seno
        py = eps * Sin[ϕ];
                   seno
        px1 = eps * Cos[ϕ1];
                    cosseno
        py1 = eps * Sin[ϕ1];
                    seno
```

However, equation 8 in De Martino establishes a relation between the momenta py and py’, thus giving us a relation between sin(ϕ) and sin(ϕ'):

sin(ϕ') = sin(ϕ) + $\frac{2d}{\epsilon l_B{}^2}$,

and since we are assuming A(x)= d = 0, then
sin(ϕ’)=sin(ϕ).
Putting this in the equation we have, the final result is as expected (see code line bellow).

```
In[•]:= d[x][ΨIII, x] - py^2 * ΨIII + eps^2 * ΨIII /. Sin[ϕ] → Sin[ϕ1] // FullSimplify // MatrixForm
                                                      seno      seno      simplifica complet···  forma de matriz
Out[•]//MatrixForm=
        (0)
        (0)
```

```
(*Region III, A(x)=-d*)
  região
```

Equation 3 with $A(x) = -d$:

$$\left[(\partial_x)^2 - p_y{}^2 + 2\,p_y\,\frac{d}{l_B{}^2} - \frac{d^2}{l_B{}^4} + \epsilon^2\right]\psi_\pm = 0$$

```
In[•]:= Remove["Global`*"]
        remove

In[•]:= d[x_] = D[#, {x, 2}] &; (*dx^2 = dx*dx but has to be applied on the function*)
                derivada

In[•]:= (*Wavefunction in region III *)
        ΨIII = t * Sqrt[px/px1] {{1}, {Exp[I * ϕ1]}} * Exp[I * px1 * x];
                   raiz quadrada      ex··· unidade im··· ex··· unidade imagina

        (*calculate equation*)
```

In[∘]:=

$$\mathtt{d[x][\Psi III,\, x] - py^2 * \Psi III + 2 * py * \frac{d}{lb^2} * \Psi III - \frac{d^2}{lb^4} * \Psi III + eps^2 * \Psi III \,//\, FullSimplify \,//\, MatrixForm}$$

simplifica completame

forma de matriz

Out[∘]//MatrixForm=

$$\begin{pmatrix} -\dfrac{e^{i\,\mathrm{px1}\,x}\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\left(\mathrm{lb}^4\left(-\mathrm{eps}^2+\mathrm{px1}^2\right)+\left(d-\mathrm{lb}^2\,\mathrm{py}\right)^2\right)t}{\mathrm{lb}^4} \\ -\dfrac{e^{i\,(\mathrm{px1}\,x+\phi 1)}\sqrt{\frac{\mathrm{px}}{\mathrm{px1}}}\left(\mathrm{lb}^4\left(-\mathrm{eps}^2+\mathrm{px1}^2\right)+\left(d-\mathrm{lb}^2\,\mathrm{py}\right)^2\right)t}{\mathrm{lb}^4} \end{pmatrix}$$

(*use the parametrization for px and py that De Martino gives*)

px = eps * Cos[ϕ]

cosseno

$$\mathtt{py = eps * Sin[\phi] + d \,/\, lb^2}$$

seno

px1 = eps * Cos[ϕ1]

cosseno

$$\mathtt{py1 = eps * Sin[\phi 1] + d \,/\, lb^2}$$

seno

Out[∘]= eps Cos[ϕ]

Out[∘]= $\dfrac{d}{\mathrm{lb}^2} + \mathrm{eps}\,\mathrm{Sin}[\phi]$

Out[∘]= eps Cos[ϕ]

Out[∘]= $\dfrac{d}{\mathrm{lb}^2} + \mathrm{eps}\,\mathrm{Sin}[\phi 1]$

In[∘]:= **(*calculate equations again*)**

$$\mathtt{d[x][\Psi III,\, x] - py^2 * \Psi III + 2 * py * \frac{d}{lb^2} * \Psi III - \frac{d^2}{lb^4} * \Psi III + eps^2 * \Psi III \,//\, FullSimplify \,//\, MatrixForm}$$

simplifica completame

forma de matriz

Out[∘]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

A.1.2 Code to Get the Transmission Amplitude Expression

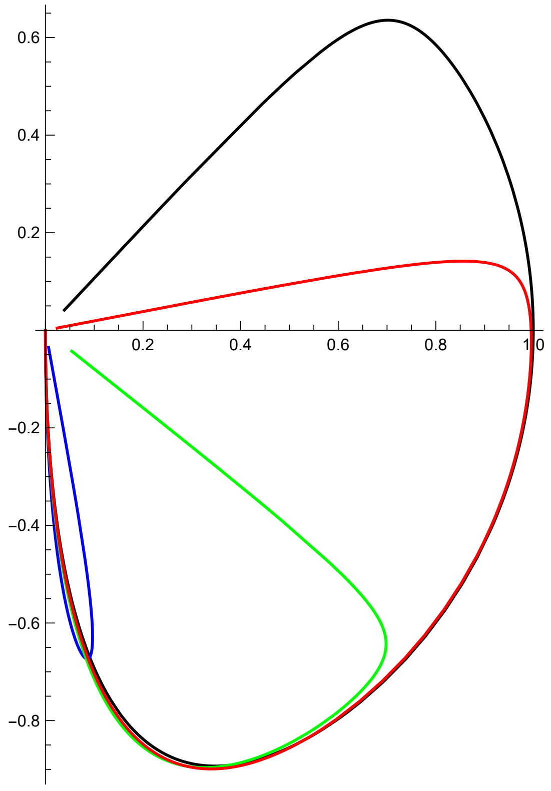
```
In[ ]:= ClearAll["Global`*"]
        apaga tudo
        ν = (((ϵ * lb)^2) / 2);
        (*z=Sqrt[2]*(py*lb); *)
            raiz quadrada
        z = Sqrt[2] * ((x / lb) + (qn * lb));
            raiz quadrada
        z1 = Sqrt[2] * (- (x / lb) - (qn * lb));
             raiz quadrada
        (* Scattering state incident from the left in the n-th mode *)
        ΦL[qn_, y_, k_, x_] = {{1}, {E^(I*ϕ)}} E^(I*k*x) + rn * {{1}, {-E^(-I*ϕ)}} E^(-I*k*x);
        Φ[qn_, y_, k_, x_] = αn * {{ParabolicCylinderD[ν - 1, z]},
                                     função D parabólica cilíndrica
              {((I * Sqrt[2]) / (ϵ * lb)) * ParabolicCylinderD[ν, z]}} +
               u·· raiz quadrada          função D parabólica cilíndrica
           βn * {{ParabolicCylinderD[ν - 1, z1]},
                  função D parabólica cilíndrica
              {- ((I * Sqrt[2]) / (ϵ * lb)) * ParabolicCylinderD[ν, z1]}};
                 u·· raiz quadrada          função D parabólica cilíndrica

        ΦR[qn_, y_, k_, x_] = tn * Sqrt[k / px1] * {{1}, {Exp[I * ϕ1]}} * Exp[I * px1 * x];
                                   raiz quadrada      ex··· unidade im··· ex··· unidade imaginária

        (* Solve equations from the continuity of Φ for tn *)
           resolve
        (*ΦL[qn,y,k1,-d] == Φ[qn2,y,k2,-d]*)
        Eq1 = FullSimplify[ΦL[qn, y, k, -d] == Φ[qn, y, k, -d]];
              simplifica completamente
        Eq2 = FullSimplify[Φ[qn, y, k, d] == ΦR[qn, y, k, d]];
              simplifica completamente
        tt = Solve[Eq1 && Eq2, {rn, tn, αn, βn}][[1]][[2]][[2]];
             resolve

        tt = FullSimplify[
             simplifica completamente
          tt /. ParabolicCylinderD[lb^2 ϵ^2 / 2, -(Sqrt[2] (d + lb^2 qn)) / lb] :→ v2m /. ParabolicCylinderD[
                função D parabólica cilíndrica                                                  função D parabólica cilíndrica
                    -1 + lb^2 ϵ^2 / 2, Sqrt[2] (d / lb + lb qn)] :→ u2p /.
               ParabolicCylinderD[lb^2 ϵ^2 / 2, Sqrt[2] (d / lb + lb qn)] :→ v2p /.
               função D parabólica cilíndrica
              ParabolicCylinderD[-1 + lb^2 ϵ^2 / 2, -(Sqrt[2] (d + lb^2 qn)) / lb] :→ u2m /.
              função D parabólica cilíndrica
             ParabolicCylinderD[lb^2 ϵ^2 / 2, Sqrt[2] (d / lb - lb qn)] :→ v1m /.
             função D parabólica cilíndrica
ParabolicCylinderD[lb^2 ϵ^2/2, √2 (-d/lb + lb qn)] :→ v1p /.
função D parabólica cilíndrica

ParabolicCylinderD[-1 + lb^2 ϵ^2/2, √2 (d/lb - lb qn)] :→ u1m /.
função D parabólica cilíndrica

ParabolicCylinderD[-1 + lb^2 ϵ^2/2, √2 (-d/lb + lb qn)] :→ u1p /.
função D parabólica cilíndrica

(-i Cos[ϕ1] + Sin[ϕ1]) :→ Exp[-I * ϕ1]]
  cosseno   seno          exp···  unidade imaginária

FullSimplify[√2 e^(-i d (k+px1)) * (1 + e^(2 i ϕ)) * 2 * i * e^(-i ϕ) * lb * (u2p v2m + u2m v2p) * ϵ] / 2;
simplifica completamente

FullSimplify[
simplifica completamente

  √(k/px1) (2 i e^(i ϕ) * (v1p v2m - v1m v2p) * 2 * i * e^(-i ϕ) + √2 * 2 * i * e^(-i ϕ) * e^(i (ϕ+ϕ1)) * lb *
     (u2p v1m + u2m v1p) * ϵ + 2 * i * e^(-i ϕ) * √2 * lb * (u1p v2m + u1m v2p) * ϵ +
    lb^2 * (u1p u2m - u1m u2p) * ϵ^2 (-i * Cos[ϕ1] + Sin[ϕ1]) * 2 * i * e^(-i ϕ))] / 2;
                                          cosseno     seno
```

Out[•]=
$$\left(\sqrt{2}\, e^{-i d (k+px1)} \left(1+e^{2 i \phi}\right) lb\,(u2p\, v2m+u2m\, v2p)\, \epsilon\right) \Big/ \left(\sqrt{\frac{k}{px1}} \left(2 i\, e^{i \phi} (v1p\, v2m-v1m\, v2p)+\sqrt{2}\, e^{i (\phi+\phi1)}\, lb\,(u2p\, v1m+u2m\, v1p)\, \epsilon + \sqrt{2}\, lb\,(u1p\, v2m+u1m\, v2p)\, \epsilon+lb^2 (u1p\, u2m-u1m\, u2p)\, \epsilon^2 (-i \operatorname{Cos}[\phi1]+\operatorname{Sin}[\phi1])\right)\right)$$

A.1.3 Code to Get the Transmission Polar Plots

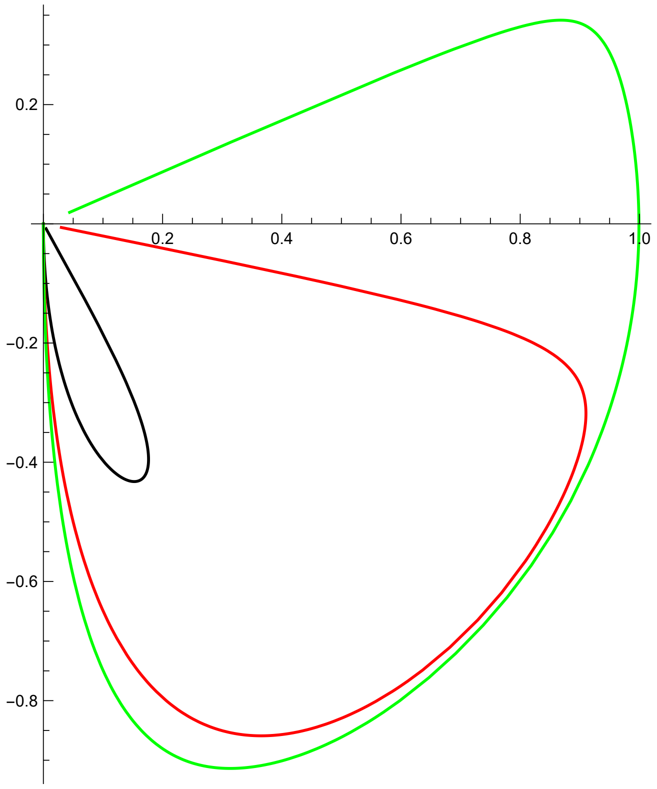
```
ClearAll["Global`*"]
apaga tudo
(*momenta parameters*)
ϕ1[ϕ_, el_, dl_] := ArcSin[2 * (dl) * (1 / (el)) + Sin[ϕ]]
                    arco seno                    seno
(*py[ϕ_]:=ϵ*Sin[ϕ]+(d/(lb)^2)*)
               seno
px[ϕ_] := Cos[ϕ]
          cosseno
px1[ϕ_, el_, dl_] := Cos[ArcSin[2 * (dl) * (1 / (el)) + Sin[ϕ]]]
                     co⋯ arco seno                    seno
(*Operators u*)
u1p[ϕ_, el_, dl_] :=
 ParabolicCylinderD[(((el)^2) / 2) - 1, Sqrt[2] * ((-dl) + el * Sin[ϕ] + dl)]
 função D parabólica cilíndrica         raiz quadrada          seno
u1m[ϕ_, el_, dl_] :=
 ParabolicCylinderD[(((el)^2) / 2) - 1, -1 * Sqrt[2] * ((-dl) + el * Sin[ϕ] + dl)]
 função D parabólica cilíndrica              raiz quadrada          seno
u2p[ϕ_, el_, dl_] :=
 ParabolicCylinderD[(((el)^2) / 2) - 1, Sqrt[2] * ((dl) + el * Sin[ϕ] + dl)]
 função D parabólica cilíndrica         raiz quadrada        seno
u2m[ϕ_, el_, dl_] :=
 ParabolicCylinderD[(((el)^2) / 2) - 1, -1 * Sqrt[2] * ((dl) + el * Sin[ϕ] + dl)]
 função D parabólica cilíndrica              raiz quadrada        seno
(*Operators v*)
v1p[ϕ_, el_, dl_] :=
 ParabolicCylinderD[((el)^2) / 2, Sqrt[2] * ((-dl) + el * Sin[ϕ] + dl)]
 função D parabólica cilíndrica   raiz quadrada          seno
v1m[ϕ_, el_, dl_] :=
 ParabolicCylinderD[((el)^2) / 2, -1 * Sqrt[2] * ((-dl) + el * Sin[ϕ] + dl)]
 função D parabólica cilíndrica        raiz quadrada          seno
v2p[ϕ_, el_, dl_] := ParabolicCylinderD[((el)^2) / 2, Sqrt[2] * ((dl) + el * Sin[ϕ] + dl)]
                     função D parabólica cilíndrica   raiz quadrada        seno
v2m[ϕ_, el_, dl_] :=
 ParabolicCylinderD[((el)^2) / 2, -1 * Sqrt[2] * ((dl) + el * Sin[ϕ] + dl)]
 função D parabólica cilíndrica        raiz quadrada        seno
(*T[ϕ_]:=(8*((3.7)^4)*Cos[ϕ1[ϕ]]*Cos[ϕ]*((u2p[ϕ]*v2m[ϕ]+v2p[ϕ]*u2m[ϕ])^2))*)
                      cosseno     cosseno
(*D operator*)
  derivada
Dop[ϕ_, el_, dl_] := ((el)^2) * Exp[I * (ϕ1[ϕ, el, dl] - ϕ)] *
                                ex⋯ unidade imaginária
    (u1p[ϕ, el, dl] * u2m[ϕ, el, dl] - u2p[ϕ, el, dl] * u1m[ϕ, el, dl]) -
   2 * (v1p[ϕ, el, dl] * v2m[ϕ, el, dl] - v2p[ϕ, el, dl] * v1m[ϕ, el, dl]) +
   I * Sqrt[2] * el * (Exp[I * ϕ1[ϕ, el, dl]] *
   u⋯ raiz quadrada    ex⋯ unidade imaginária
       (v1p[ϕ, el, dl] * u2m[ϕ, el, dl] + u2p[ϕ, el, dl] * v1m[ϕ, el, dl]) +
      Exp[-I * ϕ] * (u1p[ϕ, el, dl] * v2m[ϕ, el, dl] + v2p[ϕ, el, dl] * u1m[ϕ, el, dl]))
      exp⋯ unidade imaginária
(*transmission amplitude*)
t[ϕ_, el_, dl_] :=
  (2 * I * el * (u2p[ϕ, el, dl] * v2m[ϕ, el, dl] + v2p[ϕ, el, dl] * u2m[ϕ, el, dl])) /
      unidade imaginária
      (Exp[I * (px[ϕ] + px1[ϕ, el, dl]) * el * dl] * Dop[ϕ, el, dl])
       ⌊ex⋯ ⌊unidade imaginária

In[∘]:= (*transmission equation*)
        T[ϕ_, el_, dl_] := 2 * px1[ϕ, el, dl] * Cos[ϕ] * t[ϕ, el, dl] * (t[ϕ, el, dl])*
                                                   ⌊cosseno

In[∘]:= PolarPlot[{T[ϕ, 3.7, 0.5], T[ϕ, 3.7, 3.67], T[ϕ, 3.7, 3], T[ϕ, 3.7, 1.5]},
        ⌊gráfico polar
          {ϕ, -Pi / 2, Pi / 2}, PlotStyle → {Black, Blue, Green, Red}]
              ⌊númer⋯ ⌊número pi ⌊estilo do gráfico ⌊preto  ⌊azul  ⌊verde  ⌊vermelho
```

Out[∘]=



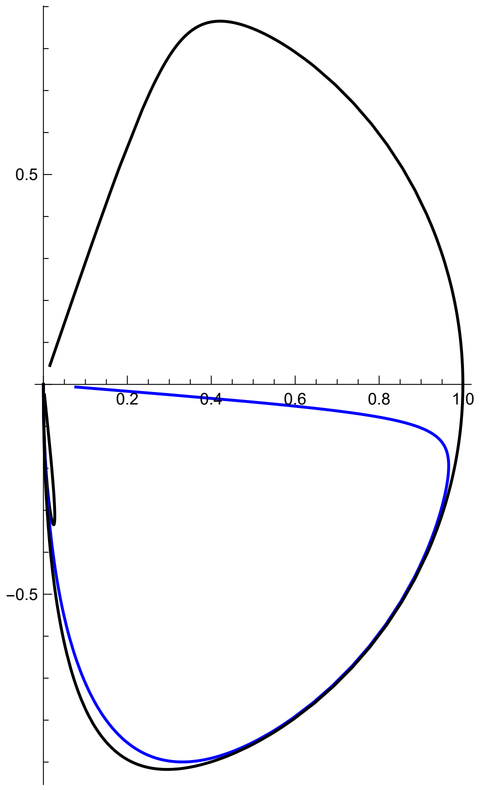
```
In[*]:= PolarPlot[{T[ϕ, 1.6, 1.5], T[ϕ, 2.5, 1.5], T[ϕ, 5, 1.5]},
        ⌊gráfico polar
          {ϕ, -Pi / 2, Pi / 2}, PlotStyle → {Black, Red, Green}]
             ⌊númer·· ⌊número pi ⌊estilo do gráfico ⌊preto ⌊ve··· ⌊verde
```

Out[*]=



```
In[ ]:= PolarPlot[{T[ϕ, 3.7, 3.69], T[ϕ, 3.7, 2.0], T[ϕ, 3.7, 0.1]},
        gráfico polar
          {ϕ, -Pi / 2, Pi / 2}, PlotStyle → {Black, Blue}]
              númer·· número pi estilo do gráfico preto    azul
```

Out[ ]=



A.2 Codes Used in Chapter 2 To Calculate the Conductivity With Magnetic Field Only

A.2.1 Parameters Mathematica Notebook

$Gr\_Vars\_Params\_21\_09\_2023.pdf$

```
In[ ]:= (************ Variables and Parameters for Calculating Conductivity of Graphene ************)

(* Constants *)
kB = 1.4×10^(-23); (* Boltzman constant (J/K) *)
e = 1.6*10^-19; (* Elementary charge (C) *)
h = 6.63* 10^(-34); (* Planck's constant (J.s) *)
ℏ = 1.05*10^(-34);  (* Planck's constant/2π (J.s) *)
hbar = ℏ;
ϵair = 8.85*10^(-12); (* Permittivity of Vacuum (F/m) *)
ϵsio2 = 3.9*8.85*10^(-12); (* Permittivity of SiO₂ (F/m) *)

a = 1.42*^-10; (* Lattice constant (m) *)
t0 = 2.7*e; (* Nearest-neighbor hopping energy (J) *)
vF = 1.0*10^6;(* Fermi velocity (m/s) *)
β = 2.5; (* Electron-phonon coupling *)
ν = 0.165;(* Poisson ratio in graphene *)

(* Variables & Parameters *)
μCont = 0.087*e;(* Contact doping (J) *)

L = 80.0*10^(-9);  (* Channel length (m) *)
W = 1000.0*10^(-9); (* Channel width (m) *)
θ = 0*π/6;    (* Crystal angle with respect to the x axis *)
dair = 50.0*10^(-9);    (* Thicknes of vacuum (m) *)
dsio2 = 0*100.0*10^(-9);   (* Thicknes of SiO2 (m) *)
cBG = (ϵair *ϵsio2)/((dsio2*ϵair) + (dair*ϵsio2)) ; (* Gate capacitance (F/m^2) *)
nimp  = 0.0*10.0^15;    (*1/m2 Density of Impurities IF nimp=0, gaussmooth must =0, if nimp≠= Gaussmooth must =1*)
temp = 0*1.5;(* Temperature of the sample (K) *)
ntmp = π/6*((kB*temp)/(hbar*vF))^2; (* Thermal carrier density *)
Rc = 0.0; (* Contact Resistance (Ohm) *)

gε = 3.0*e; (* Change in workfunction per % strain (choi2010) *)

ε0 = 0.0/100.0; (* Built-in strain *)
minε = 0.0/100.0; (* Minimum applied mechanical train *)
maxε = 0.0/100.0; (* Maximum applied mechanical train *)
Nεstep = 1; (* Strain steps *)
εStep =  (maxε - minε)/Nεstep; (* Size of strain steps *)


minB = 1; (* Minimum applied mechanical train *)
maxB = 2;(*2.05*) (* Maximum applied mechanical train *)
NBstep = 1; (* Strain steps *)
BStep =  (maxB - minB)/NBstep; (* Size of strain steps *)

Vdirac = 0.0; (* Dirac point position (V) *)
VgateMin  = 5.0; (* Minimum gate (V) *)
VgateMax = 15.0; (* Maximum gate (V) *)
Nstep  = 10;  (* Number of gate steps (V) *)
VgateStep =  (VgateMax - VgateMin)/Nstep;  (* Gate step size *)

Gausssmooth = 0; (* Boolean, smooth over discontinuity at Dirac point caused buy nimp *)

(* Arrays *)
(* Conductivity *)
Array[GG, (Nstep + 1)*(NBstep + 1) - 1];
arranjo

Array[GG1, (Nstep + 1)*(Nεstep + 1) - 1];
arranjo
Array[GG2, (Nstep + 1)*(Nεstep + 1) - 1];
arranjo
Array[GG3, (Nstep + 1)*(Nεstep + 1) - 1];
arranjo
Array[GG1p, (Nstep + 1)*(Nεstep + 1) - 1];
arranjo
Array[GG2p, (Nstep + 1)*(Nεstep + 1) - 1];
arranjo
Array[GG3p, (Nstep + 1)*(Nεstep + 1) - 1];
arranjo
```

```
(* Gate voltage, mechanical strain, fermi energy *)
Array[VVg, (Nstep + 1) * (NBstep + 1) - 1];
arranjo
Array[BB, (Nstep + 1) * (NBstep + 1) - 1];
arranjo
Array[μμ, (Nstep + 1) * (NBstep + 1) - 1];
arranjo


Array[εε, (Nstep + 1) * (Nεstep + 1) - 1];
arranjo
```

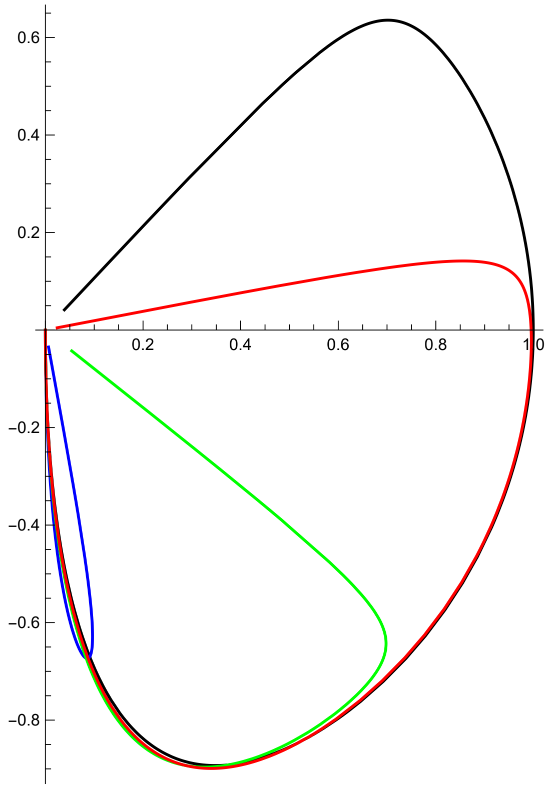
A.2.2 Code to Export the Transmission Probability

$DeMartino\_export\_Tn\_expression.pdf$

```
(* Present
 goal: to calculate the transmission for N-th mode of the transverse momentum,
                                          ⌊valor numérico
starting from the t[ϕ_,el_,dl_]
*)

In[∘]:= ClearAll["Global`*"]
        ⌊apaga tudo
(*momenta parameters*)
ϕ1[ϕ_, el_, dl_] := ArcSin[2 * (dl) * (1 / (el)) + Sin[ϕ]]
                    ⌊arco seno                   ⌊seno
(*py[ϕ_]:=ϵ*Sin[ϕ]+(d/(lb)^2)*)
             ⌊seno
px[ϕ_] := Cos[ϕ]
          ⌊cosseno
px1[ϕ_, el_, dl_] := Cos[ArcSin[2 * (dl) * (1 / (el)) + Sin[ϕ]]]
                     ⌊co⋯ ⌊arco seno                ⌊seno
(*Operators u*)
u1p[ϕ_, el_, dl_] :=
 ParabolicCylinderD[(((el)^2) / 2) - 1, Sqrt[2] * ((-dl) + el * Sin[ϕ] + dl)]
 ⌊função D parabólica cilíndrica        ⌊raiz quadrada          ⌊seno
u1m[ϕ_, el_, dl_] :=
 ParabolicCylinderD[(((el)^2) / 2) - 1, -1 * Sqrt[2] * ((-dl) + el * Sin[ϕ] + dl)]
 ⌊função D parabólica cilíndrica            ⌊raiz quadrada          ⌊seno
u2p[ϕ_, el_, dl_] :=
 ParabolicCylinderD[(((el)^2) / 2) - 1, Sqrt[2] * ((dl) + el * Sin[ϕ] + dl)]
 ⌊função D parabólica cilíndrica        ⌊raiz quadrada        ⌊seno
u2m[ϕ_, el_, dl_] :=
 ParabolicCylinderD[(((el)^2) / 2) - 1, -1 * Sqrt[2] * ((dl) + el * Sin[ϕ] + dl)]
 ⌊função D parabólica cilíndrica            ⌊raiz quadrada        ⌊seno
(*Operators v*)
v1p[ϕ_, el_, dl_] :=
 ParabolicCylinderD[((el)^2) / 2, Sqrt[2] * ((-dl) + el * Sin[ϕ] + dl)]
 ⌊função D parabólica cilíndrica  ⌊raiz quadrada          ⌊seno
v1m[ϕ_, el_, dl_] :=
 ParabolicCylinderD[((el)^2) / 2, -1 * Sqrt[2] * ((-dl) + el * Sin[ϕ] + dl)]
 ⌊função D parabólica cilíndrica      ⌊raiz quadrada          ⌊seno
v2p[ϕ_, el_, dl_] := ParabolicCylinderD[((el)^2) / 2, Sqrt[2] * ((dl) + el * Sin[ϕ] + dl)]
                     ⌊função D parabólica cilíndrica  ⌊raiz quadrada        ⌊seno
v2m[ϕ_, el_, dl_] :=
 ParabolicCylinderD[((el)^2) / 2, -1 * Sqrt[2] * ((dl) + el * Sin[ϕ] + dl)]
 ⌊função D parabólica cilíndrica      ⌊raiz quadrada        ⌊seno
(*T[ϕ_]:=(8*((3.7)^4)*Cos[ϕ1[ϕ]]*Cos[ϕ]*((u2p[ϕ]*v2m[ϕ]+v2p[ϕ]*u2m[ϕ])^2))*)
                      ⌊cosseno    ⌊cosseno
(*D operator*)
  ⌊derivada
Dop[ϕ_, el_, dl_] := ((el)^2) * Exp[I * (ϕ1[ϕ, el, dl] - ϕ)] *
                                ⌊ex⋯ ⌊unidade imaginária
    (u1p[ϕ, el, dl] * u2m[ϕ, el, dl] - u2p[ϕ, el, dl] * u1m[ϕ, el, dl]) -
   2 * (v1p[ϕ, el, dl] * v2m[ϕ, el, dl] - v2p[ϕ, el, dl] * v1m[ϕ, el, dl]) +
   I * Sqrt[2] * el * (Exp[I * ϕ1[ϕ, el, dl]] *
   ⌊u⋯ ⌊raiz quadrada   ⌊ex⋯ ⌊unidade imaginária
       (v1p[ϕ, el, dl] * u2m[ϕ, el, dl] + u2p[ϕ, el, dl] * v1m[ϕ, el, dl]) +
        Exp[-I * ϕ] * (u1p[ϕ, el, dl] * v2m[ϕ, el, dl] + v2p[ϕ, el, dl] * u1m[ϕ, el, dl]))
        ⌊exp⋯ ⌊unidade imaginária
(*transmission amplitude*)
t[ϕ_, el_, dl_] :=
 (2 * I * el * (u2p[ϕ, el, dl] * v2m[ϕ, el, dl] + v2p[ϕ, el, dl] * u2m[ϕ, el, dl])) /
      ⌊unidade imaginária
  (Exp[I * (px[ϕ] + px1[ϕ, el, dl]) * el * dl] * Dop[ϕ, el, dl])
  ⌊ex⋯ ⌊unidade imaginária


(*Sqrt[2*px1[ϕ,el,dl]/px[ϕ]]*Cos[ϕ]*)
  ⌊raiz quadrada                 ⌊cosseno
(*(2*I*3.7*Sqrt[2*px1[Pi/2,3.7,0.5]/px[Pi/2]]*Cos[Pi/2]*
     ⌊unida⋯⌊raiz quadrada ⌊número pi          ⌊número⋯ ⌊co⋯ ⌊número pi
   (u2p[Pi/2,3.7,0.5]*v2m[Pi/2,3.7,0.5]+v2p[Pi/2,3.7,0.5]*u2m[Pi/2,3.7,0.5]))
        ⌊número pi        ⌊número pi        ⌊número pi        ⌊número pi
 (Exp[I*(px[Pi/2]+px1[Pi/2,3.7,0.5])*3.7*0.5]*Dop[Pi/2,3.7,0.5])
  ⌊ex⋯⌊unida⋯⌊número pi ⌊número pi                ⌊número pi
 Sqrt[2*px1[Pi/2,3.7,0.5]/px[Pi/2]]*)
 ⌊raiz quadrada ⌊número pi     ⌊número pi
(*px1[Pi/2,3.7,0.5]
     ⌊número pi
 px[Pi/2]*)
    ⌊número pi
(*transmission equation*)
T[ϕ_, el_, dl_] := 2 * px1[ϕ, el, dl] * Cos[ϕ] * t[ϕ, el, dl] * (t[ϕ, el, dl])*
                                        ⌊cosseno
(* Clear all stored variables, but not functions *)
   ⌊apaga
Clear @@ Select[Names["Global`*"],
⌊apaga   ⌊seleci⋯ ⌊nomes de símbolos
  ToExpression[#, StandardForm, Function[sym, OwnValues[sym] =!= {}, HoldAll]] &]
  ⌊converte em expre⋯ ⌊forma padrão ⌊função ⌊valores próprios ⌊impede o cálculo de qual
(* Boudary condition in graphene for smooth edges (Tworzydolo) *)
q[N_, W_] := π / W * Sign[N] * (Abs[N] + 1 / 2)
                     ⌊fun⋯ ⌊valo⋯ ⌊val⋯ ⌊valor numérico


(*Tn[μCont_,μCH_,Nq_,L_,W_,Ay_]:=Tϕ[μCont/(ℏ*vF),μCH/(ℏ*vF),q[Nq,W],L,Ay];*)
(* Export general graphene transmission expression *)
   ⌊exporta
SetDirectory[NotebookDirectory[]];
⌊define diretório ⌊diretório do notebook


Export["Tphi_expr_V0.0.mx", T[ϕ, el, dl]]
⌊exporta

Out[•]= Tphi_expr_V0.0.mx
```

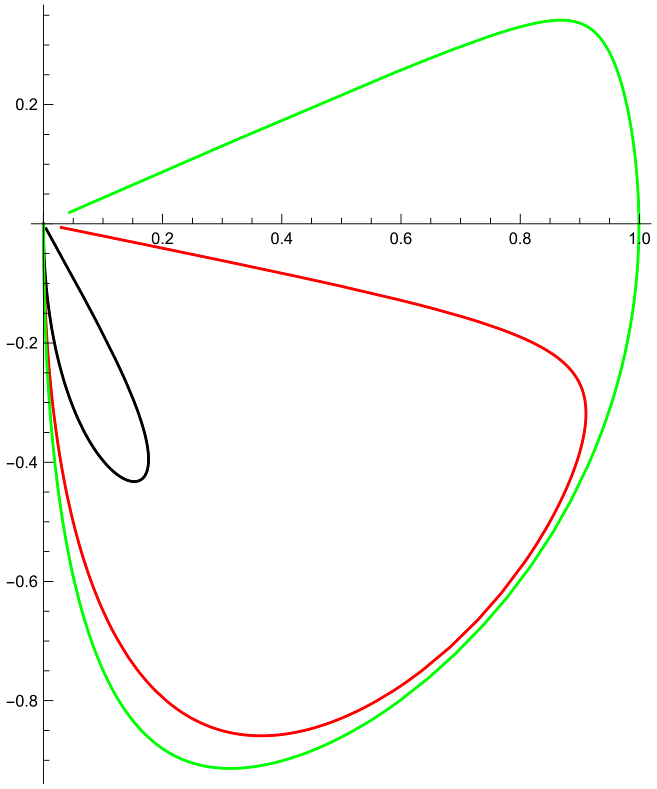
```
In[ ]:= PolarPlot[{T[ϕ, 3.7, 0.5], T[ϕ, 3.7, 3.67], T[ϕ, 3.7, 3], T[ϕ, 3.7, 1.5]},
        gráfico polar
         {ϕ, -Pi / 2, Pi / 2}, PlotStyle → {Black, Blue, Green, Red}]
              númer·· número pi estilo do gráfico preto    azul    verde    vermelho
```

Out[ ]=



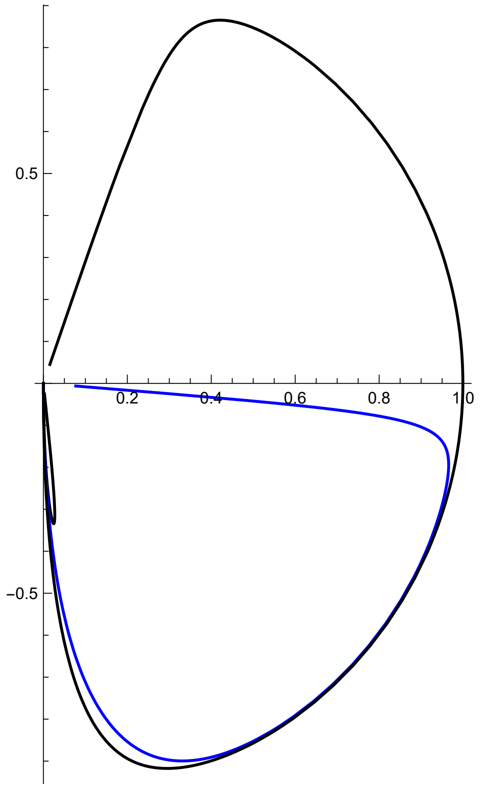
```
In[ ]:= PolarPlot[{T[ϕ, 1.6, 1.5], T[ϕ, 2.5, 1.5], T[ϕ, 5, 1.5]},
        gráfico polar
          {ϕ, -Pi / 2, Pi / 2}, PlotStyle → {Black, Red, Green}]
              númer·· número pi estilo do gráfico  preto    ve··· verde
```

Out[ ]=



```
In[ ]:= PolarPlot[{T[ϕ, 3.7, 3.69], T[ϕ, 3.7, 2.0], T[ϕ, 3.7, 0.1]},
        gráfico polar
         {ϕ, -Pi / 2, Pi / 2}, PlotStyle → {Black, Blue}]
              númer·· número pi estilo do gráfico  preto    azul
```

Out[ ]=



A.2.3 Code to Calculate Conductivity

$Conductivity\_DeMartino\_B\_VG\_ac.pdf$

```
(********** Conductivity in Graphene under a constant magnetic field **********)
ClearAll["Global`*"]    (* Clear all values and definitions *)
apaga tudo

SetDirectory[NotebookDirectory[]]; (* Set the current working directory *)
define diretório    diretório do notebook

T[ϕ_, el_, dl_] = Import["Tphi_expr_V0.0.mx"];  (* Import the transmission probability expression *)
                  importa

NotebookImport["Gr_Vars_Params_21_09_2023.nb", "Input" → "Expression"]; (* Import constants/variables/parameters *)
importa notebook

Nmax[EF_] := IntegerPart[Abs[EF]/(ħ * vF) * W/π]; (* Number of transmission modes, set by contact doping *)
             parte inteira

NBstep = 0; (*for now we are not doing a B loop*)
kFcontact = μCont / (ħ * vF);
ε = 0;
B = minB; (*need to set magnetic field here, not in parameter file*)
(*B= minB + BBStep*BStep; *) (* Incremental mechanical strain *)
lb = Sqrt[hbar / (e * B)] ;
     raiz quadrada
dl = (L / 2) / lb; (*need to calculate here for each B value, el is calculated later because it changes with Vgate*)
Ay[L_] := - (e / (hbar)) * (L / 2) * B; (* Vector potentials along y direction *)
(*Print[Ay[L]]; *)

(* For loop to increase gate voltage *)
For[Step = 0, Step ≤ (Nstep), Step += 1,
para cada

      Vgate = VgateMin + Step * VgateStep; (* Incremental gate voltage *)

      (* Carrier density from gate and scalar potential *)
      nc = cBG/e * (Vgate - Vdirac) + Sign[gε] * (gε * (1 - ν) * (ε + ε0))^2 / (π (hbar * vF)^2);
                                      função de sinal

      (* Total carrier density *)
      ntot = √(nc^2 + 4 * ((nimp/2)^2 + (ntmp)^2));

      (* Fermi level in the channel *)
      μCH = hbar * vF * If[nc ≠ 0.0, Sign[nc], 1] * √(π * Abs[ntot]) ;
                        se             função de sinal

      el =  √(π * Abs[ntot]) * lb; (*need to calculate here for each B and Vg value*)

      (* Populate array with current strain, gate, and channel fermi level *)
      VVg[Step + (NBstep) * (Nstep + 1)] = Vgate;
      BB[Step + (NBstep) * (Nstep + 1)] = B;
      μμ[Step + (NBstep) * (Nstep + 1)] = μCH ;

      (* Array for storing the average Tn *)
      Nmodes = Nmax[μCont];
   (*    Print[π/W*(Nmodes+1/2)];    *)
      Array[TnA, 2 * (Nmodes + 1), -Nmodes];
      arranjo

      (* Initialize arrays *)
      GG[Step + NBstep * (Nstep + 1)]  = 0;

      (*For loop to sum over all transmission modes*)
      For[Nq = -Nmodes, Nq ≤ (Nmodes + 1), Nq += 1,
      para cada

       qn = π / W * (Nq - 1 / 2) + Ay[L];
       ϕ = ArcSin[qn / kFcontact] ;
           arco seno
```

```
            TnA[Nq] = Re[T[ϕ, el, dl]];
                      parte real

            (* Sum over transmission modes to get conductance *)
            GG[Step + (NBstep) * (Nstep + 1)] += 4*e^2/h * TnA[Nq] * (4*e^2/(Pi*h))^(-1);

               ];

      ];
      (*Build table of data:  Gate voltage, conductivity, mechanical strain, Fermi energy in the channel*)
      TB1 = Table[{VVg[n], Re[GG[n]] * L / W, BB[n], μμ[n] / e}, {n, 0, (Nstep + 1) * (NBstep + 1) - 1, 1}];
            tabela          parte real
      TB2 = Prepend[TB1, {"Vg", "Sig", "B", "EF"}]; (* Add headers to columns *)
            adiciona no início


      Export["G_DeMartino_14112023_B1_10Vg_points.dat" , TB2, "Table" ]
      exporta

Out[ ]= G_DeMartino_14112023_B1_10Vg_points.dat
```

A.3 Codes Used in Chapter 3 To Calculate the Conductivity With Magnetic Field and Strain

This Section contains the codes used to calculate G in graphene with magnetic field and strain. We calculate for two cases: with constant values of B and varying the values of ϵ and with constant values of strain and varying the values of B. As mentioned before, to improve computation time, we did not write a code to do both loops at the same time.

A.3.1 Parameters for Constant B and Varying Strain

```
(************ Variables and Parameters for Calculating Conductivity of Graphene with B fixed************)

(* Constants *)
kB = 1.4×10^(-23); (* Boltzman constant (J/K) *)
e = 1.6*10^-19; (* Elementary charge (C) *)
h = 6.63* 10^(-34); (* Planck's constant (J.s) *)
ħ = 1.05*10^(-34);  (* Planck's constant/2π (J.s) *)
hbar = ħ;
ϵair = 8.85*10^(-12); (* Permittivity of Vacuum (F/m) *)
ϵsio2 = 3.9*8.85*10^(-12); (* Permittivity of SiO₂ (F/m) *)

a = 1.42*^-10; (* Lattice constant (m) *)
t0 = 2.7*e; (* Nearest-neighbor hopping energy (J) *)
vF = 1.0*10^6;(* Fermi velocity (m/s) *)
β = 2.5; (* Electron-phonon coupling *)
ν = 0.165;(* Poisson ratio in graphene *)

(* Variables & Parameters *)
μCont = 0.087*e;(* Contact doping (J) *)

L = 80.0*10^(-9);  (* Channel length (m) *)
W = 1000.0*10^(-9); (* Channel width (m) *)
θ = 15.0*π/6;     (* Crystal angle with respect to the x axis *)
dair = 50.0*10^(-9);     (* Thicknes of vacuum (m) *)
dsio2 = 0*100.0*10^(-9);   (* Thicknes of SiO2 (m) *)
cBG = (ϵair *ϵsio2)/((dsio2*ϵair) + (dair*ϵsio2)) ; (* Gate capacitance (F/m^2) *)
nimp  = 0.0*10.0^15;    (*1/m2 Density of Impurities IF nimp=0, gaussmooth must =0, if nimp≠= Gaussmooth must =1*)
temp = 0*1.5;(* Temperature of the sample (K) *)
ntmp = π/6*((kB*temp)/(hbar*vF))^2; (* Thermal carrier density *)
Rc = 0.0; (* Contact Resistance (Ohm) *)

gε = 3.0*e; (* Change in workfunction per % strain (choi2010) *)

ε0 = 0.0/100.0; (* Built-in strain *)
minε = 1.9/100.0; (* Minimum applied mechanical train *)
maxε = 1.99/100.0; (* Maximum applied mechanical train *)
Nεstep = 1; (* Strain steps *)
εStep =  (maxε - minε)/Nεstep; (* Size of strain steps *)

minB = 1.0; (* Minimum applied B *)
maxB = 2.03;(*2.05*) (* Maximum applied B *)
NBstep = 1; (* Strain steps *)
BStep =  (maxB - minB)/NBstep; (* Size of B steps *)

Vdirac = 0.0; (* Dirac point position (V) *)
VgateMin  = 5.0; (* Minimum gate (V) *)
VgateMax = 15.0; (* Maximum gate (V) *)
Nstep  = 10;  (* Number of gate steps (V) *)
VgateStep =  (VgateMax - VgateMin)/Nstep;  (* Gate step size *)

Gausssmooth = 0;  (* Boolean, smooth over discontinuity at Dirac point caused buy nimp *)

(* Arrays *)
(* Conductivity *)
Array[GG, (Nstep + 1) * (Nεstep + 1) - 1];
arranjo

(* Gate voltage, mechanical strain, fermi energy *)
Array[VVg, (Nstep + 1) * (Nεstep + 1) - 1];
arranjo
Array[BB, (Nstep + 1) * (Nεstep + 1) - 1];
arranjo
Array[μμ, (Nstep + 1) * (Nεstep + 1) - 1];
arranjo
Array[εε, (Nstep + 1) * (Nεstep + 1) - 1];
arranjo
```

A.3.2 Parameters for Constant Strain and Varying B

```
(************ Variables and Parameters for Calculating Conductivity of Graphene with strain fixed************)

(* Constants *)
kB = 1.4×10^(-23); (* Boltzman constant (J/K) *)
e = 1.6*10^-19; (* Elementary charge (C) *)
h = 6.63* 10^(-34); (* Planck's constant (J.s) *)
ħ = 1.05*10^(-34);  (* Planck's constant/2π (J.s) *)
hbar = ħ;
ϵair = 8.85*10^(-12); (* Permittivity of Vacuum (F/m) *)
ϵsio2 = 3.9*8.85*10^(-12); (* Permittivity of SiO2 (F/m) *)

a = 1.42*^-10; (* Lattice constant (m) *)
t0 = 2.7*e; (* Nearest-neighbor hopping energy (J) *)
vF = 1.0*10^6;(* Fermi velocity (m/s) *)
β = 2.5; (* Electron-phonon coupling *)
ν = 0.165;(* Poisson ratio in graphene *)

(* Variables & Parameters *)
μCont = 0.087*e;(* Contact doping (J) *)

L = 80.0*10^(-9);  (* Channel length (m) *)
W = 1000.0*10^(-9); (* Channel width (m) *)
θ = 15.0*π/6;     (* Crystal angle with respect to the x axis *)
dair = 50.0*10^(-9);     (* Thicknes of vacuum (m) *)
dsio2 = 0*100.0*10^(-9);   (* Thicknes of SiO2 (m) *)
cBG = (ϵair *ϵsio2)/((dsio2*ϵair) + (dair*ϵsio2)) ; (* Gate capacitance (F/m^2) *)
nimp  = 0.0*10.0^15;    (*1/m2 Density of Impurities IF nimp=0, gaussmooth must =0, if nimp≠= Gaussmooth must =1*)
temp = 0*1.5;(* Temperature of the sample (K) *)
ntmp = π/6*((kB*temp)/(hbar*vF))^2; (* Thermal carrier density *)
Rc = 0.0; (* Contact Resistance (Ohm) *)

gε = 3.0*e; (* Change in workfunction per % strain (choi2010) *)

ε0 = 0.0/100.0; (* Built-in strain *)
minε = 1.0/100.0; (* Minimum applied mechanical train *)
maxε = 1.9/100.0; (* Maximum applied mechanical train *)
Nεstep = 1; (* Strain steps *)
εStep =  (maxε - minε)/Nεstep; (* Size of strain steps *)

minB = 1.0; (* Minimum applied B *)
maxB = 2.03;(*2.05*) (* Maximum applied B *)
NBstep = 10; (* Strain steps *)
BStep =  (maxB - minB)/NBstep; (* Size of B steps *)

Vdirac = 0.0; (* Dirac point position (V) *)
VgateMin  = 5.0; (* Minimum gate (V) *)
VgateMax = 15.0; (* Maximum gate (V) *)
Nstep  = 10;  (* Number of gate steps (V) *)
VgateStep =  (VgateMax - VgateMin)/Nstep;  (* Gate step size *)

Gausssmooth = 0;  (* Boolean, smooth over discontinuity at Dirac point caused buy nimp *)

(* Arrays *)
(* Conductivity *)
Array[GG, (Nstep + 1) * (NBstep + 1) - 1];
arranjo

(* Gate voltage, mechanical strain, fermi energy *)
Array[VVg, (Nstep + 1) * (NBstep + 1) - 1];
arranjo
Array[BB, (Nstep + 1) * (NBstep + 1) - 1];
arranjo
Array[μμ, (Nstep + 1) * (NBstep + 1) - 1];
arranjo
Array[εε, (Nstep + 1) * (NBstep + 1) - 1];
arranjo
```

A.3.3 Code to Export the Transmission Probability

```
(* Present goal: update the functions to include the effect of strain,
and get the new transmission amplitude expression. Then test it to see if it makes
 physical sense. My current understanding is that phi, phi1, px, and px1,
which are only evaluated at the edges of the chann∈l, do not change. *)

In[∘]:= ClearAll["Global`*"]
        apaga tudo
        (*momenta parameters*)
        ϕ1[ϕ_, ∈l_, dl_] := ArcSin[2 * (dl) * (1 / (∈l)) + Sin[ϕ]];
                            arco seno                       seno
        (*py[ϕ_]:=∈*Sin[ϕ]+(d/(lb)^2)*)
                       seno
        px[ϕ_] := Cos[ϕ];
                  cosseno
        px1[ϕ_, ∈l_, dl_] := Cos[ArcSin[2 * (dl) * (1 / (∈l)) + Sin[ϕ]]];
                             co⋯ arco seno                      seno

        (*The effect of strain is simply to change the value of dl in u1 and v1
          from -d/l to (-d-xo)/l and in u2 and v2 from d/l to (d-xo)/l*)

        (*Operators u*)
        u1p[ϕ_, ∈l_, dl_] :=
          ParabolicCylinderD[(((∈l)^2) / 2) - 1, Sqrt[2] * ((-dl) + ∈l * Sin[ϕ] + dl)];
          função D parabólica cilíndrica         raiz quadrada          seno
        u1m[ϕ_, ∈l_, dl_] :=
         ParabolicCylinderD[(((∈l)^2) / 2) - 1, -1 * Sqrt[2] * ((-dl) + ∈l * Sin[ϕ] + dl)]
         função D parabólica cilíndrica              raiz quadrada          seno
        u2p[ϕ_, ∈l_, dl_] :=
         ParabolicCylinderD[(((∈l)^2) / 2) - 1, Sqrt[2] * ((dl) + ∈l * Sin[ϕ] + dl)]
         função D parabólica cilíndrica         raiz quadrada        seno
        u2m[ϕ_, ∈l_, dl_] :=
         ParabolicCylinderD[(((∈l)^2) / 2) - 1, -1 * Sqrt[2] * ((dl) + ∈l * Sin[ϕ] + dl)]
         função D parabólica cilíndrica              raiz quadrada        seno
        (*Operators v*)
        v1p[ϕ_, ∈l_, dl_] :=
         ParabolicCylinderD[((∈l)^2) / 2, Sqrt[2] * ((-dl) + ∈l * Sin[ϕ] + dl)]
         função D parabólica cilíndrica    raiz quadrada          seno
        v1m[ϕ_, ∈l_, dl_] :=
         ParabolicCylinderD[((∈l)^2) / 2, -1 * Sqrt[2] * ((-dl) + ∈l * Sin[ϕ] + dl)]
         função D parabólica cilíndrica         raiz quadrada          seno
        v2p[ϕ_, ∈l_, dl_] := ParabolicCylinderD[((∈l)^2) / 2, Sqrt[2] * ((dl) + ∈l * Sin[ϕ] + dl)]
                             função D parabólica cilíndrica    raiz quadrada        seno
        v2m[ϕ_, ∈l_, dl_] :=
         ParabolicCylinderD[((∈l)^2) / 2, -1 * Sqrt[2] * ((dl) + ∈l * Sin[ϕ] + dl)]
         função D parabólica cilíndrica         raiz quadrada        seno

        (*D operator without strain*)
          derivada
        Dop[ϕ_, ∈l_, dl_] := ((∈l)^2) * Exp[I * (ϕ1[ϕ, ∈l, dl] - ϕ)] *
                                        ex⋯ unidade imaginária
            (u1p[ϕ, ∈l, dl] * u2m[ϕ, ∈l, dl] - u2p[ϕ, ∈l, dl] * u1m[ϕ, ∈l, dl]) -
          2 * (v1p[ϕ, ∈l, dl] * v2m[ϕ, ∈l, dl] - v2p[ϕ, ∈l, dl] * v1m[ϕ, ∈l, dl]) +
    I * Sqrt[2] * ϵl * (Exp[I * ϕ1[ϕ, ϵl, dl]] *
   ⌊u·· ⌊raiz quadrada      ⌊ex··· ⌊unidade imaginária
        (v1p[ϕ, ϵl, dl] * u2m[ϕ, ϵl, dl] + u2p[ϕ, ϵl, dl] * v1m[ϕ, ϵl, dl]) +
      Exp[-I * ϕ] * (u1p[ϕ, ϵl, dl] * v2m[ϕ, ϵl, dl] + v2p[ϕ, ϵl, dl] * u1m[ϕ, ϵl, dl]))
      ⌊exp··· ⌊unidade imaginária

(*transmission amplitude without strain*)
t[ϕ_, ϵl_, dl_] :=
 (2 * I * ϵl * (u2p[ϕ, ϵl, dl] * v2m[ϕ, ϵl, dl] + v2p[ϕ, ϵl, dl] * u2m[ϕ, ϵl, dl])) /
      ⌊unidade imaginária
  (Exp[I * (px[ϕ] + px1[ϕ, ϵl, dl]) * ϵl * dl] * Dop[ϕ, ϵl, dl])
   ⌊ex··· ⌊unidade imaginária


(*transmission equation without strain*)
T[ϕ_, ϵl_, dl_] := 2 * px1[ϕ, ϵl, dl] * Cos[ϕ] * t[ϕ, ϵl, dl] * (t[ϕ, ϵl, dl])*
                                         ⌊cosseno


(* Export general graphene transmission expression *)
   ⌊exporta
SetDirectory[NotebookDirectory[]];
⌊define diretório  ⌊diretório do notebook
Export["Tphi_expr_V0.0.mx", T[ϕ, ϵl, dl]]
⌊exporta

"Tphi_expr_V0.0.mx"
```

```
Out[•]= Tphi_expr_V0.0.mx

Out[•]= Tphi_expr_V0.0.mx
```

```
In[ ]:= (*D operator with strain*)
         ⌊derivada

        Dops[ϕ_, ϵl_, dl_, dl1_, dl2_] := ((ϵl)^2) * Exp[I * (ϕ1[ϕ, ϵl, dl] - ϕ)] *
                                                        ⌊ex··· ⌊unidade imaginária
           (u1p[ϕ, ϵl, dl1] * u2m[ϕ, ϵl, dl2] - u2p[ϕ, ϵl, dl2] * u1m[ϕ, ϵl, dl1]) -
          2 * (v1p[ϕ, ϵl, dl1] * v2m[ϕ, ϵl, dl2] - v2p[ϕ, ϵl, dl2] * v1m[ϕ, ϵl, dl1]) +
          I * Sqrt[2] * ϵl * (Exp[I * ϕ1[ϕ, ϵl, dl]] *
         ⌊u·· ⌊raiz quadrada   ⌊ex··· ⌊unidade imaginária
              (v1p[ϕ, ϵl, dl1] * u2m[ϕ, ϵl, dl2] + u2p[ϕ, ϵl, dl2] * v1m[ϕ, ϵl, dl1]) +
             Exp[-I * ϕ] * (u1p[ϕ, ϵl, dl1] * v2m[ϕ, ϵl, dl2] + v2p[ϕ, ϵl, dl2] * u1m[ϕ, ϵl, dl1]))
            ⌊exp···  ⌊unidade imaginária

        (*transmission amplitude with strain*)
        ts[ϕ_, ϵl_, dl_, dl1_, dl2_] :=
         (2 * I * ϵl * (u2p[ϕ, ϵl, dl2] * v2m[ϕ, ϵl, dl2] + v2p[ϕ, ϵl, dl2] * u2m[ϕ, ϵl, dl2])) /
             ⌊unidade imaginária
          (Exp[I * (px[ϕ] + px1[ϕ, ϵl, dl]) * ϵl * dl] * Dops[ϕ, ϵl, dl, dl1, dl2])
           ⌊ex··· ⌊unidade imaginária

        (*transmission equation with strain*)
        Ts[ϕ_, ϵl_, dl_, dl1_, dl2_] :=
         2 * px1[ϕ, ϵl, dl] * Cos[ϕ] * ts[ϕ, ϵl, dl, dl1, dl2] * (ts[ϕ, ϵl, dl, dl1, dl2])*
                              ⌊cosseno
        SetDirectory[NotebookDirectory[]];
        ⌊define diretório  ⌊diretório do notebook
        Export["Tsphi_expr_V0.mx", Ts[ϕ, ϵl, dl, dl1, dl2]];
        ⌊exporta
```

A.3.4 Code to Calculate Conductivity for Constant B and Varying Strain

```
(********** Conductivity in Graphene under Uniaxial Strain and Magnetic Field - Fixed B and Varying Strain **********)
ClearAll["Global`*"]    (* Clear all values and definitions *)
apaga tudo

SetDirectory[NotebookDirectory[]]; (* Set the current working directory *)
define diretório    diretório do notebook

Ts[ϕ_, ϵL_, dl_, dl1_, dl2_] = Import["Tsphi_expr_V0.mx"];  (* Import the transmission probability expression *)
                                importa

NotebookImport["Params_strain_Vg_withB_Nov_2023.nb", "Input" → "Expression"]; (* Import constants/variables/parameters *)
importa notebook

Nmax[EF_] := IntegerPart[ Abs[EF]/(ħ * vF) * W/π + 0.5]; (* Number of transmission modes, set by contact doping *)
             parte inteira

kFcontact = μCont / (ħ * vF) ;

(* For loop to increase magnetic field *) For[SStep = 0, SStep ≤ (Nεstep), SStep += 1,
                                         para cada

  B = minB; (* setting B field *)
  lb = Sqrt[hbar / (e * B)] ;
       raiz quadrada
  ε = minε + SStep * εStep;

  (* Incremental mechanical strain *)

  (* Magnetic Vector potentials along y direction *)
  Ay[L_] := - (e / (hbar)) * (L / 2) * B;

  (* Strain Vector potentials along y direction *)
  Ay1[θ_] := (4 π (ε + ε0))/(3 √3 a) ν Sin[θ] + (β (ε + ε0) (1 + ν))/(2 a) Sin[3 * θ];
                                   seno                                  seno

  Ay2[θ_] := (2 π (ε + ε0))/(3 a) (-(1/√3) ν Sin[θ] + ν Cos[θ]) + (β (ε + ε0) (1 + ν))/(2 a) Sin[3 * θ];
                                                seno      cosseno                                seno

  Ay3[θ_] := (2 π (ε + ε0))/(3 a) (-(1/√3) ν Sin[θ] - ν Cos[θ]) + (β (ε + ε0) (1 + ν))/(2 a) Sin[3 * θ];
                                                seno      cosseno                                seno


  (*xo =A0 (elB^2/c) , and A0=(hbar/e)*A_strain , so this means that xo is different in all 3 valley, since A_strain is different in each valley
    so define x1=(hbar/e)*(elB^2/c)*Ay1, etc.  *)
  (*Now remember than DeMartino has different units than us,and in our units,
  we replace their c by hbar.Such that in our code,A0(elB^2/c)  =A0(elB^2/hbar)=A_strain*(elB^2/hbar)*(hbar/e)=A_strain*lB^2*)
  x1 = (lb)^2 * Ay1[θ];
  x2 = (lb)^2 * Ay2[θ];
  x3 = (lb)^2 * Ay3[θ];
  dl  = (L / 2) / lb; (*need to calculate here for each B value, el is calculated later because it changes with Vgate*)
  (*because of strain dl is now different on left side = dll, so define dll1, dll2, dll3*)
  dll1  = ((L / 2) + x1) / lb;
  dll2  = ((L / 2) + x2) / lb;
  dll3 = ((L / 2) + x3) / lb;
  (*because of strain dl is now different on right side = dlr, so define dlr1, dlr2, dlr3*)
  dlr1  = ((L / 2) - x1) / lb;
  dlr2  = ((L / 2) - x2) / lb;
  dlr3  = ((L / 2) - x3) / lb;

  (* For loop to increase gate voltage *)
  For[Step = 0, Step ≤ (Nstep), Step += 1,
  para cada

      Vgate = VgateMin + Step * VgateStep;  (* Incremental gate voltage *)

      (* Carrier density from gate and scalar potential *)
    nc = cBG/e * (Vgate - Vdirac) + Sign[gε] * (gε * (1 - ν) * (ε + ε0))^2 / (π (hbar * vF)^2);
                                    função de sinal

    (* Total carrier density *)
    ntot = Sqrt[nc^2 + 4 * (((nimp/2)^2 + (ntmp)^2)];

    (* Fermi level in the channel *)
    μCH = hbar * vF * If[nc ≠ 0.0, Sign[nc], 1] * Sqrt[π * Abs[ntot]];
                      se            função de sinal

    el = Sqrt[π * Abs[ntot]] * lb; (*need to calculate here for each B and Vg value*)

    (* Populate array with current strain, gate, and channel fermi level *)
    VVg[Step + (SStep) * (Nstep + 1)] = Vgate;
    BB[Step + (SStep) * (Nstep + 1)] = B;
    μμ[Step + (SStep) * (Nstep + 1)] = μCH;
    εε[Step + (SStep) * (Nstep + 1)] = ε;

    (* Array for storing the average Tn *)
    Nmodes = Nmax[μCont];
    Array[TnA, 2 * (Nmodes + 1), -Nmodes];
    arranjo

    (* Initialize arrays *)
    GG[Step + SStep * (Nstep + 1)] = 0;

    (*For loop to sum over all transmission modes*)
    For[Nq = -Nmodes, Nq ≤ (Nmodes + 1), Nq += 1,
    para cada

     qn1n = π / W * (Nq - 1 / 2) + Ay[L] - Ay1[θ];
     qn2n = π / W * (Nq - 1 / 2) + Ay[L] - Ay2[θ];
     qn3n = π / W * (Nq - 1 / 2) + Ay[L] - Ay3[θ];
     qn1p = π / W * (Nq - 1 / 2) + Ay[L] + Ay1[θ];
     qn2p = π / W * (Nq - 1 / 2) + Ay[L] + Ay2[θ];
     qn3p = π / W * (Nq - 1 / 2) + Ay[L] + Ay3[θ];
     ϕ1n = ArcSin[qn1n / kFcontact] ;
           arco seno
     ϕ2n = ArcSin[qn2n / kFcontact] ;
           arco seno
     ϕ3n = ArcSin[qn3n / kFcontact] ;
           arco seno
     ϕ1p = ArcSin[qn1p / kFcontact] ;
           arco seno
     ϕ2p = ArcSin[qn2p / kFcontact] ;
           arco seno
     ϕ3p = ArcSin[qn3p / kFcontact] ;
           arco seno

      (* Average transmission for all 6 Dirac cones - modified to only keep positive values, and replace negative values by zeros,
  basically using the fact that in experiment we have a bias voltage focing current in one direction only*)

  TnA1 = Re[Ts[ϕ1n, el, dl, dll1, dlr1]];
         parte real
  If[TnA1 ≥ 0, , TnA1 = 0.0];
  se
  TnA2 = Re[Ts[ϕ2n, el, dl, dll2, dlr2]];
         parte real
  If[TnA2 ≥ 0, , TnA2 = 0.0];
  se
  TnA3 = Re[Ts[ϕ3n, el, dl, dll3, dlr3]];
         parte real
  If[TnA3 ≥ 0, , TnA3 = 0.0];
  se
  TnA4 = Re[Ts[ϕ1p, el, dl, dlr1, dll1]];
         parte real
  If[TnA4 ≥ 0, , TnA4 = 0.0];
  se
```

```
    TnA5 = Re[Ts[ϕ2p, el, dl, dlr2, dll2]];
           parte real
    If[TnA5 ≥ 0, , TnA5 = 0.0];
    se
    TnA6 = Re[Ts[ϕ3p, el, dl, dlr3, dll3]];
           parte real
    If[TnA6 ≥ 0, , TnA6 = 0.0];
    se

    TnA[Nq] = (TnA1 + TnA2 + TnA3 + TnA4 + TnA5 + TnA6) / 6;

       (* Sum over transmission modes to get conductance *)
```

$$GG[Step + (SStep) * (Nstep + 1)] += \frac{4 * e\verb|^|2}{h} * TnA[Nq] * \left(\frac{4 * e\verb|^|2}{Pi * h}\right)\verb|^|(-1);$$

```
          ];
       ];
  ];


(*Build table of data:  Gate voltage, conductivity, magnetic field,  strain, Fermi energy in the channel*)
TB1 = Table[{VVg[n], Re[GG[n]] * L / W, BB[n], εε[n], μμ[n] / e}, {n, 0, (Nstep + 1) * (Nεstep + 1) - 1, 1}];
      tabela            parte real
(* Add headers to columns *)
TB2 = Prepend[TB1, {"Vg", "Sig", "B", "Strain", "EF"}];
      adiciona no início

Export["test14022024_1Tfield_1p9to1p99strain.dat" , TB2, "Table" ]
exporta
```

```
Out[ ]= test14022024_1Tfield_1p9to1p99strain.dat
```

A.3.5 Code to Calculate Conductivity for Constant Strain and Varying B

```
(********** Conductivity in Graphene under Uniaxial Strain and Magetic Field - Fixed Strain and Varying Magnetic Field **********)
ClearAll["Global`*"]   (* Clear all values and definitions *)
apaga tudo

SetDirectory[NotebookDirectory[]]; (* Set the current working directory *)
define diretório   diretório do notebook

Ts[ϕ_, ϵl_, dl_, dl1_, dl2_] = Import["Tsphi_expr_V0.mx"];  (* Import the transmission probability expression *)
                               importa

NotebookImport["Params_B_Vg_with_strain_Nov_2023.nb", "Input" → "Expression"]; (* Import constants/variables/parameters *)
importa notebook

Nmax[EF_] := IntegerPart[(Abs[EF] / (ℏ * vF)) * (W / π)  + 0.5]; (* Number of transmission modes, set by contact doping *)
             parte inteira  valor absoluto

kFcontact = μCont / (ℏ * vF) ;

(* For loop to increase magnetic field *) For[BBstep = 0, BBstep ≤ (NBstep), BBstep += 1,
                                          para cada

  B = minB + BBstep * BStep; (* Incrementing B field *)
  lb = Sqrt[hbar / (e * B)] ;
       raiz quadrada
  ε = minε;

  (* Magnetic Vector potentials along y direction *)
  Ay[L_] := - (e / (hbar)) * (L / 2) * B;

  (* Strain Vector potentials along y direction *)
  Ay1[θ_] := ((4 π (ε + ε0)) / (3 × √3 a)) ν Sin[θ] + ((β (ε + ε0) (1 + ν)) / (2 a)) Sin[3 * θ];
                                             seno                                seno
  Ay2[θ_] := ((2 π (ε + ε0)) / (3 a)) (-(1/(√3)) ν Sin[θ] + ν Cos[θ]) + ((β (ε + ε0) (1 + ν)) / (2 a)) Sin[3 * θ];
                                                   seno       cosseno                                  seno
  Ay3[θ_] := ((2 π (ε + ε0)) / (3 a)) (-(1/(√3)) ν Sin[θ] - ν Cos[θ]) + ((β (ε + ε0) (1 + ν)) / (2 a)) Sin[3 * θ];
                                                   seno       cosseno                                  seno

  (*x_o =A_0 (el_B^2/c) , and A_0=(hbar/e)*A_strain , so this means that x_o is different in all 3 valley, since A_strain is different in each valley
    so define x_1=(hbar/e)*(el_B^2/c)*Ay1, etc.  *)
  (*Now remember than DeMartino has different units than us,and in our units,
  we replace their c by hbar.Such that in our code,A_0(el_B^2/c)  =A_0(el_B^2/hbar)=A_strain*(el_B^2/hbar)*(hbar/e)=A_strain*l_B^2*)
  x1 = (lb) ^2 * Ay1[θ];
  x2 = (lb) ^2 * Ay2[θ];
  x3 = (lb) ^2 * Ay3[θ];
  dl  = (L / 2) / lb; (*need to calculate here for each B value, el is calculated later because it changes with Vgate*)
  (*because of strain dl is now different on left side = dll, so define dll1, dll2, dll3*)
  dll1  = ((L / 2) + x1) / lb;
  dll2  = ((L / 2) + x2) / lb;
  dll3 = ((L / 2) + x3) / lb;
  (*because of strain dl is now different on right side = dlr, so define dlr1, dlr2, dlr3*)
  dlr1  = ((L / 2) - x1) / lb;
  dlr2  = ((L / 2) - x2) / lb;
  dlr3  = ((L / 2) - x3) / lb;

  (* For loop to increase gate voltage *)
  For[Step = 0, Step ≤ (Nstep), Step += 1,
  para cada

      Vgate = VgateMin + Step * VgateStep;  (* Incremental gate voltage *)

      (* Carrier density from gate and scalar potential *)
      nc = (cBG / e) * (Vgate - Vdirac) + Sign[gε] * ((gε * (1 - ν) * (ε + ε0))^2 / (π (hbar * vF)^2));
                                          função de sinal

      (* Total carrier density *)
      ntot = √(nc^2 + 4 * ((nimp / 2) ^2 + (ntmp) ^2));

      (* Fermi level in the channel *)
      μCH = hbar * vF * If[nc ≠ 0.0, Sign[nc], 1] * √(π * Abs[ntot]);
                        se           função de sinal   valor absoluto
```

```
          el = √(π*Abs[ntot]) *lb; (*need to calculate here for each B and Vg value*)
                   valor absoluto

          (* Populate array with current strain, gate, and channel fermi level *)
          VVg[Step + (BBstep) * (Nstep + 1)] = Vgate;
          BB[Step + (BBstep) * (Nstep + 1)] = B;
          μμ[Step + (BBstep) * (Nstep + 1)] = μCH ;
          εε[Step + (BBstep) * (Nstep + 1)] = ε;

          (* Array for storing the average Tn *)
          Nmodes = Nmax[μCont];
          Array[TnA, 2* (Nmodes + 1), -Nmodes];
          arranjo

          (* Initialize arrays *)
          GG[Step + BBstep* (Nstep + 1)] = 0;

          (*For loop to sum over all transmission modes*)
          For[Nq = -Nmodes, Nq ≤ (Nmodes + 1), Nq += 1,
          para cada

           qn1n = π/W* (Nq - 1/2) + Ay[L] - Ay1[θ];
           qn2n = π/W* (Nq - 1/2) + Ay[L] - Ay2[θ];
           qn3n = π/W* (Nq - 1/2) + Ay[L] - Ay3[θ];
           qn1p = π/W* (Nq - 1/2) + Ay[L] + Ay1[θ];
           qn2p = π/W* (Nq - 1/2) + Ay[L] + Ay2[θ];
           qn3p = π/W* (Nq - 1/2) + Ay[L] + Ay3[θ];
           ϕ1n = ArcSin[qn1n/kFcontact] ;
                 arco seno
           ϕ2n = ArcSin[qn2n/kFcontact] ;
                 arco seno
           ϕ3n = ArcSin[qn3n/kFcontact] ;
                 arco seno
           ϕ1p = ArcSin[qn1p/kFcontact] ;
                 arco seno
           ϕ2p = ArcSin[qn2p/kFcontact] ;
                 arco seno
           ϕ3p = ArcSin[qn3p/kFcontact] ;
                 arco seno

           (* Average transmission for all 6 Dirac cones - modified to only keep positive values, and replace negative values by zeros,
       basically using the fact that in experiment we have a bias voltage focing current in one direction only*)

       TnA1 = Re[Ts[ϕ1n, el, dl, dll1, dlr1]];
              parte real
       If[TnA1 ≥ 0, , TnA1 = 0.0];
       se
       TnA2 = Re[Ts[ϕ2n, el, dl, dll2, dlr2]];
              parte real
       If[TnA2 ≥ 0, , TnA2 = 0.0];
       se
       TnA3 = Re[Ts[ϕ3n, el, dl, dll3, dlr3]];
              parte real
       If[TnA3 ≥ 0, , TnA3 = 0.0];
       se
       TnA4 = Re[Ts[ϕ1p, el, dl, dlr1, dll1]];
              parte real
       If[TnA4 ≥ 0, , TnA4 = 0.0];
       se
       TnA5 = Re[Ts[ϕ2p, el, dl, dlr2, dll2]];
              parte real
       If[TnA5 ≥ 0, , TnA5 = 0.0];
       se
       TnA6 = Re[Ts[ϕ3p, el, dl, dlr3, dll3]];
              parte real
       If[TnA6 ≥ 0, , TnA6 = 0.0];
       se

       TnA[Nq] = (TnA1 + TnA2 + TnA3 + TnA4 + TnA5 + TnA6) / 6;

           (* Sum over transmission modes to get conductance *)
           GG[Step + (BBstep) * (Nstep + 1)] += ((4*e^2) / h) * TnA[Nq] * ((4*e^2) / (Pi*h)) ^ (-1);
                                                                                número pi

              ];
```

```
        ];
    ];


  (*Build table of data:  Gate voltage, conductivity, magnetic field,  strain, Fermi energy in the channel*)
  TB1 = Table[{VVg[n], Re[GG[n]] * L / W, BB[n], εε[n], μμ[n] / e}, {n, 0, (Nstep + 1) * (NBstep + 1) - 1, 1}];
        tabela             parte real
  (* Add headers to columns *)
  TB2 = Prepend[TB1, {"Vg", "Sig", "B", "Strain", "EF"}];
        adiciona no início

  Export["piru.dat" , TB2, "Table" ]
  exporta

Out[]= piru.dat

In[]:= "G_DeMartino_strain_24112023_B_Vg_s1.dat"

Out[]= G_DeMartino_strain_24112023_B_Vg_s1.dat
```

Bibliography

[1] A. C. McRae, G. Wei, and A. R. Champagne. Graphene quantum strain transistors. *Phys. Rev. Appl.*, 11(5):054019, 2019.

[2] A. De Martino, L. Dell'Anna, and R. Egger. Magnetic confinement of massless dirac fermions in graphene. *Phys. Rev. Lett.*, 98(6):066802, 2007.

[3] J. Tworzydło, B. Trauzettel, M. Titov, A. Rycerz, and C. W. J. Beenakker. Sub-poissonian shot noise in graphene. *Phys. Rev. Lett.*, 96(24):246802, 2006.

[4] A. H. Castro Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov, and A. K. Geim. The electronic properties of graphene. *Rev. Mod. Phys.*, 81(1):109–162, 2009.

[5] F. Guinea, M. I. Katsnelson, and A. K. Geim. Energy gaps and a zero-field quantum hall effect in graphene by strain engineering. *Nature Phys.*, 6(1):30–33, 2010.

[6] A. Bhagat and K. Mullen. Pseudo-magnetic quantum hall effect in oscillating graphene. *Solid State Communications*, 287:31–34, 2019.

[7] A. J. M. Giesbers, L. A. Ponomarenko, K. S. Novoselov, A. K. Geim, M. I. Katsnelson, J. C. Maan, and U. Zeitler. Gap opening in the zeroth landau level of graphene. *Phys. Rev. B.*, 80(20):201403, 2009. PRB.

[8] Mikkel Settnes, Jose H. Garcia, and Stephan Roche. Valley-polarized quantum transport generated by gauge fields in graphene. *2D Mater.*, 4(3):031006, 2017.

[9] D. Grassano, M. D'Alessandro, O. Pulci, S. G. Sharapov, V. P. Gusynin, and A. A. Varlamov. Work function, deformation potential, and collapse of landau levels in strained graphene and silicene. *Phys. Rev. B.*, 101(24):245115, 2020.

[10] Gerardo G. Naumis, Salvador Barraza-Lopez, Maurice Oliva-Leyva, and Humberto Terrones. Electronic and optical properties of strained graphene and other strained 2d materials: a review. *Rep. Prog. Phys.*, 80(9):096501, 2017.

[11] Miloud Mekkaoui, Ahmed Jellal, and Hocine Bahlouli. Tunneling of electrons in graphene via double triangular barrier in external fields. *Solid State Commun.*, 358:114981, 2022.

[12] A. Zubarev and D. Dragoman. Ballistic charge carrier transmission through graphene multi-barrier structures in uniform magnetic field. *J. Phys. D: Appl. Phys.*, 47(42):425302, 2014.

[13] Kenneth S. Burch, David Mandrus, and Je-Geun Park. Magnetism in two-dimensional van der waals materials. *Nature*, 563(7729):47–52, 2018.

[14] P. Nigge, A. C. Qu, É Lantagne-Hurtubise, E. Mårsell, S. Link, G. Tom, M. Zonno, M. Michiardi, M. Schneider, S. Zhdanovich, G. Levy, U. Starke, C. Gutiérrez, D. Bonn, S. A. Burke, M. Franz, and A. Damascelli. Room temperature strain-induced landau levels in graphene on a wafer-scale platform. *Sci. Adv.*, 5(11):eaaw5593, 2019.

[15] Arnab Pal, Shuo Zhang, Tanmay Chavan, Kunjesh Agashiwala, Chao-Hui Yeh, Wei Cao, and Kaustav Banerjee. Quantum-engineered devices based on 2d materials for next-generation information processing and storage. *Adv. Mater.*, 35(27):2109894, 2023.

[16] Ya-Ning Ren, Qiang Cheng, Si-Yu Li, Chao Yan, Yi-Wen Liu, Ke Lv, Mo-Han Zhang, Qing-Feng Sun, and Lin He. Spatial and magnetic confinement of massless dirac fermions. *Phys. Rev. B.*, 104(16):L161408, 2021.

[17] Fengcheng Wu and Sankar Das Sarma. Identification of superconducting pairing symmetry in twisted bilayer graphene using in-plane magnetic field and strain. *Phys. Rev. B.*, 99(22):220507, 2019.

[18] M. M. Fogler, F. Guinea, and M. I. Katsnelson. Pseudomagnetic fields and ballistic transport in a suspended graphene sheet. *Phys. Rev. Lett.*, 101(22):226804, 2008.

[19] A. C. McRae, G. Wei, L. Huang, Serap Yiğen, Vahid Tayari, and A R Champagne. Mechanical control of quantum transport in graphene. 2023.

[20] Feng Miao, Shi-Jun Liang, and Bin Cheng. Straintronics with van der waals materials. *Npj Quantum Mater.*, 6(1):59, 2021.

[21] Seon-Myeong Choi, Seung-Hoon Jhi, and Young-Woo Son. Effects of strain on electronic properties of graphene. *Phys. Rev. B.*, 81(8):081407, 2010.

[22] Peter Rickhaus, Romain Maurand, Ming-Hao Liu, Markus Weiss, Klaus Richter, and Christian Schönenberger. Ballistic interferences in suspended graphene. *Nat. Commun.*, 4(1):2342, 2013.

[23] I. Solomonovich Gradshteyn and Iosif Moiseevich Ryzhik. *Table of integrals, series, and products*. Academic Press, 2014.

[24] Piranavan Kumaravadivel, Scott Mills, and Xu Du. Magnetic field suppression of andreev conductance at superconductor–graphene interface. *2D Mater.*, 4(4):045011, 2017.

[25] Alexander V. Savin, Yuri S. Kivshar, and Bambi Hu. Suppression of thermal conductivity in graphene nanoribbons with rough edges. *Phys. Rev. B*, 82:195422, 2010.

[26] Alex Zazunov, Arijit Kundu, Artur Hütten, and Reinhold Egger. Magnetic scattering of dirac fermions in topological insulators and graphene. *Phys. Rev. B.*, 82(15):155431, 2010.

[27] Si Wu, Matthew Killi, and Arun Paramekanti. Graphene under spatially varying external potentials: Landau levels, magnetotransport, and topological modes. *Phys. Rev. B.*, 85(19):195404, 2012.

[28] Yingjie Zhang, Youngseok Kim, Matthew J. Gilbert, and Nadya Mason. Magnetotransport in a strain superlattice of graphene. *Appl. Phys. Lett.*, 115(14):143508, 2019.

[29] Fengnian Xia, Damon B Farmer, Yu-ming Lin, and Phaedon Avouris. Graphene field-effect transistors with high on/off current ratio and large transport band gap at room temperature. *Nano letters*, 10(2):715–718, 2010.